ENGLISCH · GERMER
SCHEUSE · THRUN

CPC 464
TIPS & TRICKS

EINE FUNDGRUBE FÜR DEN
CPC 464 ANWENDER

EIN DATA BECKER BUCH

ENGLISCH · GERMER
SCHEUSE · THRUN

CPC 464

TIPS & TRICKS

EINE FUNDGRUBE FÜR DEN CPC 464 ANWENDER

EIN DATA BECKER BUCH

I S B N 3-89011-039-8

INHALTSVERZEICHNIS

KAPITEL 4 MASCHINENSPRACHE

KAPITEL 5 BASIC-ZEILEN VARIABLE UND TOKENS

KAPITEL 6 NÜTZLICHE ROUTINEN



KAPITEL 1 VORWORT

Als wir das Angebot bekamen, dieses Buch zu schreiben, waren wir natürlich Feuer und Flamme.

Dieses um so mehr, als wir die ersten Daten unseres neuen Computers hörten. Als wir ihn dann vor ca. 3 Monaten zum ersten Mal "in der Hand" hielten und die ersten paar Stunden mit ihm gearbeitet hatten, waren wir hellauf begeistert.

Nach einiger Zeit wurde uns klar, daß dieser Computer wohl als derjenige bezeichnet werden kann, auf den viele gewartet haben.

Er ist gleichzeitig geeignet für den Schüler, den Mathematiker, den Einsteiger, den Aufsteiger, den Experten, den kommerziell Interessierten, den Basicprogrammierer, den Graphikfreak, den Soundversessenen usw., usw..

Das breite Anwendungsspektrum spiegelt sich wieder in der Vielzahl der Themenbereiche, die in diesem Buch angesprochen werden. Um zu beweisen, daß alles nicht nur graue Theorie ist, haben wir dieses Buch mit möglichst vielen (nachvollziehbaren) Beispielprogrammen gespickt.

Das einzige, was wir Ihnen noch mit auf den Weg geben möchten, wenn Sie dieses Buch lesen:
Denken Sie daran, ein Computer ist Ihr guter Kamerad, behandeln Sie ihn dementsprechend pfleglich, er wird es Ihnen danken (unser ist in der ganzen Zeit kein einziges Mal (!) unbeabsichtigt ausgestiegen).

Düsseldorf, im August 1984

Die Autoren

KAPITEL 2 GRAPHIK

2.1 EINFÜHRUNG

Die Graphik ist eines der faszinierensten Teile eines Computers. Darum haben die Entwickler des CPC auch solch einen Wert auf diesen Teil Ihres Computers gelegt. Ihr CPC 464 hat jetzt folgende Graphikmöglichkeiten:

Auflösung von 640*200 Punkten
(=128000 Einzelpunkte)
20-40-80 Zeichen bei 25 Zeilen
16-4-2 Farben gleichzeitig darstellbar
32 verschiedene Farben
Leistungsfähige Graphikbfehle
Windowtechnologie
Selbstdefinierte Zeichen
Graphikzeichen

Die Entwicklung im Graphikbereich hat die Computer erst zu dem universellen Instrument gemacht, daß sie heute sind. Der Mensch ist ein optisch orientiertes Wesen, so ist eine Reihe von Zahlen für einen Menschen nicht allzu leicht zu verstehen, bekommt er diese Zahlen aber graphisch aufbereitet angezeigt, so ist die Information viel schneller und leichter aufzunehmen. Da das Ergebnis von Computerberechnungen viele Zahlen sind, ist eine graphische Aufbereitung nötig. Die Hilfsmittel und Möglichkeiten, die Ihr CPC in diesem Bereich hat, stellen wir Ihnen in diesem Kapitel ausführlich vor.

Wir untermalen unsere theoretischen Erklärungen, wie immer mit praktischen Beispielen. Sie sollten aber auch selbst ausprobieren und experimentieren.

2.2 DIE GRAPHIKAUFLÖSUNG - GRAPHIKEN ZUM ANSCHAUEN

Die Auflösung ist ein Schlagwort, mit dem alle Hersteller versuchen, ihren Computer aufzuwerten. Was ist aber nun diese Auflösung? Um Ihnen dieses zu verdeutlichen, nehmen wir ein Beispiel aus der Zeit, als noch Bleistift und Papier die Welt regierten. Es ist ganz natürlich, daß Sie mit einem feinen Bleistift weitaus bessere Zeichnungen anfertigen, als mit einem dicken. Genauso ist es mit der Auflösung. Durch eine höhere Auflösung ist es möglich, feinere Graphiken zu erstellen. Dieses wird möglich durch eine höhere Anzahl von Einzelpunkten, die Sie ansprechen können. Bei Ihrem CPC haben Sie ein Raster von 640 Punkten in horizontale und 200 Punkten in vertikale Richtung, somit können Sie also 128000 Punkte setzen (leuchten lassen). Hiermit können Sie wirklich gute Balken, Torten oder Liniengraphiken machen. Natürlich können Sie auch Bilder zeichnen, Spiele entwerfen und vieles mehr.

Im folgenden haben wir Ihnen vier Programme geschrieben, die Ihnen die Graphikfähigkeiten Ihres neuen Computers verdeutlichen. Tippen Sie die Beispiele ab und lassen sich dann mittels RUN in die Wunderwelt der Graphik entführen. Sollten Ihnen die Beispiele langweilig werden, verändern Sie die Programme oder programmieren Sie einfach ganz neue. Doch bevor Sie dieses tun, sollten Sie sich besser erst einmal dieses Kapitel bis zum Ende durchlesen.

2.2.1 GRAPHIKEN ZUM ANSCHAUEN

Der folgende Teil dieses Unterkapitels besteht aus kleinen Routinen, die Graphiken erstellen. Sie können sie nur anschauen, aber auch in Ihre Programme einbinden.

Das erste Programm zeichnet einen Kreis. Nach Eingabe von Ursprung, Radius und Schrittweite. Sie haben die Wahl, ob nur der Kreis gezeichnet werden soll, oder ob dieser, bzw. der Hintergrund farblich ausgefüllt werden sollen. Bei zu großer Schrittweite ist es möglich, daß nicht der ganze Kreis (Hintergrund) ausgefüllt wird bzw.. Wählen Si deshalb die Schrittweite immer möglichst klein.

```
10 MODE 2
20 INPUT"Ursprung (X)";x
30 INPUT"Ursprung (Y)";y
40 INPUT"Radius (R)";r
50 INPUT"Schrittweite";s
60 INPUT "'H'intergrund oder 'K'reis ausfuellen";a$
70 CLS
80 z=(1 AND a$="h")+(2 AND a$="k")
90 ORIGIN x,y
100 FOR n=1 TO 360 STEP s
110 ex=ex+1
120 xp=r*COS(n)
130 yp=r*SIN(n)
140 PLOT xp,yp,1
150 IF yp=ABS(yp) THEN ey=400-y ELSE ey=y-400
160 IF xp=ABS(xp) THEN ex=640-x ELSE ex=x-640
170 IF z=1 THEN DRAW xp,ey
180 IF z=1 THEN PLOT xp,yp:DRAW ex,yp
190 IF z=2 THEN MOVE 0,0: DRAW xp,yp
200 NEXT
210 IF z<>1 THEN GOTO 280
220 FOR n=r TO 640-x
230 MOVE n,0:DRAW n,400-y
240 MOVE -n,0:DRAW -n,400-y
250 MOVE n,0:DRAW n,y-400
260 MOVE -n,0:DRAW -n,y-400
270 NEXT
280 a$=INKEY$:IF a$="" THEN 280
290 RUN
```

```
Zum Beispiel: Ursprung (x)=320
              Ursprung (z)=200
              Radius      =30
              Schrittweite=0.5
```

Als zweite Anschauungsgraphik haben wir eine pseudodreidimensionale Graphik uns ausgesucht. Aus der Zeichnung kann man viele Bilder erkennen, zum Beispiel eine Pyramide von oben, oder ein Gang in die Tiefe oder auch ein Spinnennetz, je nach Ihrer Vorstellungsgabe. Schon durch kleine Veränderung des Programms entsteht ein neues Bild.

```
10 MODE 2
20 ORIGIN 320,200
30 FOR t=0 TO 180 STEP 10
40 PLOT a,t:DRAW t,-a
50 PLOT-a,t:DRAW-t,-a
60 PLOT 180,180:DRAW 0,0
70 PLOT -180,180:DRAW 0,0
80 PLOT a,t:DRAW -a,t
90 PLOT 180,-180:DRAW 0,0
100 PLOT -180,-180:DRAW 0,0
110 PLOT-t,-a:DRAW t,-a
120 a=a+10
130 NEXT t
```

Wenn Sie zum Beispiel die Zeile 30 umändern in:

```
30 FOR T=180 TO 0 STEP -10,
```

so erhalten Sie ein ganz anderes Bild
Experimentieren Sie einmal mit diesem Programm, Sie werden verblüffende Ergebnisse erzielen.

Die dritte Graphik wird Kaleidoskop genannt, nach dem Programmstart werden Sie verstehen warum. Hier ist auch der Zufallsfaktor im Spiel. Mit ihm lassen sich sehr interessante Graphiken entwickeln.

```
10 MODE 2
20 DEF FN f(x)=INT(RND(1)*x)+1
30 FOR x=1 TO 400 STEP FN f(25)+5
40 PLOT x,0:DRAW 400,x
50 PLOT 0,x:DRAW x,400
60 PLOT x,0:DRAW 0,400-x
70 PLOT 400,x:DRAW 400-x,400
80 PLOT x,0:DRAW 400-x,400
90 PLOT 400,x:DRAW 0,400-x
100 NEXT
110 LOCATE 55,1
120 PRINT "Noch einmal?"
130 a$=INKEY$:IF a$="" THEN 130
140 IF a$="j" THEN RUN
150 END
```

Die vierte Graphik ist ein Sinusrundfächer. Das Programm zeichnet nach Eingabe einer Schrittweite eine Sinuskurve, jedoch nicht mit gesetzten Punkten, sondern mit Linien von einem Punkt (0,200) zum aktuellen Punkt der Sinuskurve. Das Bild, das entsteht, ist wieder pseudodreidimensional. Die optische Täuschung entsteht durch die Überlagerung der Linien.

```
10 REM Sinusrundfaecher
20 DEG
30 CLG
40 INPUT"Schrittweite ";s
50 DEF FN f(x)=SIN(X)
60 FOR n=1 to 640 STEP s
70 x=x+(s*1.125)
80 MOVE 0,200
90 DRAW n,200+FN f(x)*200
100 NEXT n
110 a$=INKEY$:IF a$="" THEN 110
120 RUN
```

Versuchen Sie, dieses Programm auch einmal mit COS(X), mit -COS(X) oder mit -SIN(X).

Natürlich können Sie die hohe Auflösung des CPC nicht nur für sogenannte Computer-Art (Kunst mit aus dem Computer) benutzen. So lassen sich zum Beispiel mit Hilfe eines Programmes Zeichnungen erstellen, abspeichern, laden und so

weiter. Sie können sich Ihr eigenes Wappen am Bildschirm entwerfen. Wenn Sie sich dann noch eine Druckroutine zurechtbasteln, können Sie dieses dann ausdrucken und eingerahmt an die Türe Ihres "Schlosses" hängen (Natürlich gibt es auch sinnvollere Anwendungen).

Oder sollten Sie stark mathematisch interessiert sein? Dann wäre es Ihnen sicherlich eine große Hilfe, wenn Ihnen Ihr Computer eine Funktion graphisch ausgeben würde. Wie Sie sicher wissen, läßt sich am Graphen schon einiges abschätzen, was man sonst mühsam berechnen muß.

Für den Fall, daß Sie selbst einmal programmieren möchten (z.B. kommerzielle Programme) kommt Ihnen die hohe Auflösung natürlich auch zugute. Möchten Sie z.B. die Zuwachsraten Ihres Geschäftes der letzten zwölf Monate graphisch ausgeben, so ist das mit dem CPC ein Kinderspiel.

All diese schönen Anwendungen wären nicht zu realisieren, hätte Ihr Computer eine geringe Auflösung oder schwache Graphikbefehle, wie es leider bei einigen anderen Computern der Fall ist.

2.2.2 LISSAJOUS-FIGUREN

Jules Antoin Lissajous, ein französischer Physiker, der von 1822 bis 1880 lebte, machte eine Studie der Bewegung von Teilchen unter dem Einfluß periodischer Bewegungen. Er entdeckte, daß sich Körper unter dem Einfluß dieser Bewegungen in verschiedenen Kurven bewegen.
Wir haben ein Programm geschrieben, das Ihnen das Aussehen dieser Kurven zeigen soll.
Nach Start des Programmes müssen Sie drei Eingaben machen:
1.) Y-Frequenz; diese sollte im Bereich von 0 bis 20 liegen
2.) X-Frequenz; diese sollte im Bereich von 0 bis 20 liegen
3.) Schrittzahl; dieser Wert gibt an, wieviele Punkte gesetzt werden.

```
10 REM Lissajous Figuren
20 MODE 0
30 INPUT "Y-Frequenz";y
40 INPUT "X-Frequenz";x
50 INPUT"Schrittzahl";s
60 CLS
70 FOR a=0 TO 2*PI STEP 2*PI/s
80 PLOT 150*SIN(a*x)+150,100*COS(a*y)+100,1
90 NEXT
100 END
15 DEG
```

Erklärung zum Listing:

10	Titel
15	Umstellung des Rechensystems auf Bogenmaß
20	Einstellen des Bildschirmmodus
30	Eingabe der Y-Frequenz
40	Eingabe der X-Frequenz
50	Eingabe der Schrittzahl
60	Löschen des Bildschirms
70	Eröffnen einer Schleife
80	Setzen der Punkte
90	Schließen der Schleife
100	Programmende

2.3 GRAPHIKEDITOR

Zur Benutzung des Programms benötigen Sie einen Atari kompatiblen Joystick
Nach dem Start des Programmes können Sie einen kleinen Punkt, Ihren Graphikcursor, mit dem Joystick über den Bildschirm bewegen. Nach dem Drücken des Feuerknopfes können Sie Linien ziehen. Um diese Linien zu löschen, müssen Sie die Spacetaste drücken und mit dem Graphikcursor über die Linien fahren. Möchten Sie den ganzen Bildschirm löschen, so betätigen Sie die "C" Taste. Nach Abschluß der Zeichnung können Sie mit der "S" Taste das Bild abspeichern. Sie werden gefragt, wie das Bild heißen soll und nach Eingabe des Namens werden Sie aufgefordert, die Tasten "REC" und "PLAY" zu drücken. Danach wird die Graphik, Block für Block, abgespeichert. Nach Beendigung des Speichervorganges werden Sie gefragt, ob Sie weiter zeichnen oder das Programm beenden möchten. Um ein Bild vom Programm aus zu laden, drücken Sie die "L" Taste. Sie werden aufgefordert, die "PLAY" Taste zu drücken. Nun wird das nächste Bild auf Ihrer eingelegten Kassette geladen. Wenn Sie ein Bild ohne den Grafikeditor laden möchten, so geben Sie folgendes ein:

```
10 WINDOW 1,79,24,25:LOAD""
```

Starten Sie nun mit "RUN" und Ihr Bild wird eingeladen.
Dies bewirkt, daß die Grafik nicht von der Schrift zerstört wird und der Ausdruck der Anweisungen nur in den letzten beiden Zeilen erfolgt.

Noch ein Hinweis zum Schluß, wenn Sie das Programm beendet haben, geben Sie den Befehl "Mode 2" ein. Dieser hebt die Fenster wieder auf, die vorher definiert worden sind und löscht die Graphik.

```
10 MODE 2
20 CLS
30 GOSUB 240
40 x=1:y=1:z=0
50 PLOT x,y,1
60 jo=JOY(0):IF jo<>0 THEN 80
70 a$=INKEY$:IF A$="" THEN 70
80 PLOT x,y,z
90 y=y+(1 AND jo=1)-(1 AND jo=2)
100 IF y<0 THEN y=0
110 IF y>367 THEN y=367
120 x=x+(1 AND jo=8)-(1 AND jo=4)
130 IF jo=9 THEN x=x+1:y=y+1
140 IF jo=10 THEN x=x+1:y=y-1
150 IF jo=6 THEN x=x-1:y=y-1
160 IF jo=5 THEN x=x-1:y=y+1
170 IF x<0 THEN x=0
180 IF x>639 THEN x=639
190 IF z=0 AND a$=" " THEN z=1:GOTO 210
200 IF z=1 AND a$=" " THEN z=0
210 IF a$="e" THEN GOTO 280
220 IF a$="c" THEN CLS
230 GOTO 50
240 REM
250 ORIGIN 0,33
260 WINDOW #0,1,79,24,25
270 RETURN
280 WINDOW SWAP 0
290 INPUT"Wie soll das Bild heissen?",a$
300 SPEED WRITE 1
310 SAVE a$,b,49152,16383
320 PRINT"Neue 'G'rafik oder 'E'nde"
330 in$=INKEY$:IF in$="g" THEN RUN
340 IF in$<>"e" THEN GOTO 330
```

2.4 FUNKTIONENPLOTTER

Ein Funktionenplotter ist eines der wichtigsten Hilfsmittel eines Mathematikers, kann aber auch jedem Schüler bei seinen Mathematikhausaufgaben hilfreich sein. Der folgende Plotter ermöglicht es Ihnen, eine Funktion in Zeile 30 einzutragen,, die nach Eingabe des X Bereiches geplottet wird. Zum Programmablauf: Zuerst werden Eingaben benötigt, und zwar der Minimalwert für X, der Maximalwert für X und die Schrittweite. Nach diesen Angaben wird das Maximum bzw. das Minimum der Funktionswerte annähernd berechnet, um die Y-Achse maßstabsgerecht zu zeichnen. Dies geschieht in der Unterroutine ab 370, hier werden zwanzig Werte getestet, um den maximalen bzw. minimalen Y-Wert zu errechnen. Nach dem Rücksprung wird das Koordinatensystem gezeichnet und benannt. Nun wird die Funktion gezeichnet. Zuerst der negative Ast, dann der positive. Nach Beendigung der Zeichnung wartet das Programm so lange bis Sie eine Taste drücken.

Die angewandte Funktion ergibt eine Parabel, deren negativer Teil über die X-Achse geklappt ist. Versuchen Sie einmal als Minimum (-2), als Maximum (+2) und als Schrittweite (0.1). Nach kurzer Rechenzeit wird eine recht anschauliche Kurve geplottet.

Verwenden Sie bitte als Variable immer den Buchstaben X.

```
10 REM ***************** FUNKTIONSPLOTTER *****************
20 MODE 2
30 CLEAR
40 DEG
50 DEF FN f(x)=ABS(x^2-1)
60 INPUT"minimum (x)";a
70 INPUT"maximum (x)";b
80 INPUT"Schrittweite";c
90 IF a>b THEN 20
100 GOSUB 370
```

```
110 CLG
120 PLOT 0,200:DRAW 640,200:PLOT 320,0:DRAW 320,400
130 LOCATE 1,14
140 mp=320/ABS(a)
150 mq=320/ABS(b)
160 IF mp=mq THEN pr=a
170 IF mp<mq THEN pr=a
180 IF mp>mq THEN pr=-b
190 PRINT pr
200 pr$=STR$(ABS(pr))
210 LOCATE (80-LEN (pr$)),14
220 PRINT pr$
230 de$=STR$(de(1))
240 LOCATE 39-LEN (de$),1
250 PRINT de$
260 de$=STR$(-de(1))
270 LOCATE 39-LEN(de$),25
280 PRINT de$
290 mn=200/de(1)
300 FOR s=0.1 TO a STEP -c
310 PLOT 320+s*mp,200+FN f(s)*mn,1
320 NEXT
330 FOR n=0.1 TO b STEP c
340 PLOT 320+n*mq,200+FN f(n)*mn
350 NEXT
360 a$=INKEY$:IF a$="" THEN 360 ELSE 20
370 ml =(ABS(a)+ABS(b))
380 det=ml/20
390 ml=ml+2
400 DIM de(22)
410 FOR i=a TO b STEP det
420 var=var+1
430 de(var)=(FN f(i))
440 NEXT i
450 de(21)=(FN f(a)):de(22)=(FN F(b))
460 FOR j=21 TO 1 STEP -1
470 FOR i=1 TO j
480 IF ABS(de(i+1))<ABS(de(i)) THEN 520
490 de=de(i+1)
500 de(i+1)=de(i)
510 de(i)=de
520 NEXT i
530 NEXT j
540 PRINT"Wahrscheinliches Maximum bzw.Minumum der Funktion
in diesem Bereich:",de(1)
550 de(1)=ABS(de(1))
560 PRINT"Taste druecken"
570 a$=INKEY$:IF a$="" THEN 570
580 RETURN
```

2.5 FARBEN UND ZEICHEN

Ihr CPC ist auf dem Gebiet der Farben und Zeichen ebenfalls als Supercomputer zu bezeichnen.

Leider ist die höchste Farbzahl (32) nicht in jedem Modus erreichbar. Achten Sie bitte beim Programmieren auf diese Einschränkung, da Sie sich vielleicht sonst über ein IMPROPER ARGUMENT wundern werden. Wieviele Farben Sie in welchem Modus verwenden können, ist in Ihrem Handbuch gut erklärt. Lesen Sie diese Abschnitte bitte gut durch.

Die verschiedenen Modi (20,40,80 Zeichen) werden im Handbuch ebenfalls leicht verständlich erläutert. Achten Sie auch auf diese Kapitel.

Wir führen Ihnen hier noch einmal explizit die Befehle für die Modi und Farbeinstellung auf, da diese im Handbuch an keiner Stelle tabellarisch aufgeführt sind.

Mode 0	Löscht den Bildschirm und schaltet auf 20 Zeichen-Modus
Mode 1	Löscht den Bildschirm und schaltet auf 40 Zeichen-Modus
Mode 2	Löscht den Bildschirm und schaltet auf 80 Zeichen-Modus
Border	Legt die Farbe des Randes fest (bis zu 26). Werden 2 Farben angegeben, wechselt die Rahmenfarbe zwischen diesen beiden
Ink	Ordnet einem bestimmten Kanal (erster Wert) eine Farbe zu (zweiter Wert).

Pen	Ordnet bei der Schrift einem Kanal (erster Wert) eine Farbe zu (zweiter Wert).
Paper	Ordnet beim Hintergrund einem Kanal (erster Wert) eine Farbe zu (zweiter Wert).

2.6 LEISTUNGSFÄHIGE GRAPHIKBEFEHLE

Im folgenden Kapitel werden die nächsten Befehle erklärt:

PLOT und PLOTR
DRAW und DRAWR
MOVE und MOVER
ORIGIN
TEST und TESTR
XPOS und YPOS

Der erste Befehl in unserer Liste heißt PLOT X,Y und setzt einen Punkt an die absolute Koordinate X,Y. Absolut heißt in diesem Fall, daß die Position des Puntes nur abhängig vom aktuellen Ursprung ist. Im Normalfall liegt dieser Ursprung bei 0,0 in der linken, unteren Bildschirmecke. PLOTR X,Y setzt ebenfalls einen Punkt, jedoch relativ an die Koordinate X,Y. Relativ heißt, daß der X und der Y-Wert relativ zur Graphikcursorposition berechnet werden.

Ein sehr wichtiger Befehl ist DRAW X,Y. Dieser Befehl zieht eine Linie von der Cursorposition zur absoluten Koordinate X,Y. DRAWR zieht ebenfalls eine Linie zur Koordinate X,Y, jedoch werden auch hier die Koordinaten relativ zur Graphikcursorposition berechnet.

Um den Graphikcursor zu bewegen, ohne einen Punkt zu setzen, wird der Befehl MOVE X,Y benutzt. Er bewegt den Cursor zur Koordinate X,Y absolut zum Ursprung. Bei MOVER X,Y wird der Cursor an die, zur alten Position relativ berechneten, Koordinate X,Y bewegt.

ORIGIN ist der Befehl zum Umstellen des Ursprunges, von dem aus die absoluten Koordinaten berechnet werden. Weiterhin kann mit dem ORIGIN-Befehl ein Graphikfenster aufgebaut werden. Das Originalfenster mißt 640 Punkte in der Breite und 200 Punkte in der Höhe.

Mit dem Befehl TEST X,Y können Sie die Farbe des Punktes an der Koordinate X,Y auslesen. Auch bei diesem Befehl besteht die Möglichkeit, die Koordinate relativ zu rechnen, dann heißt der Befehl TESTR.

Um die aktuelle Graphikcursorposition zu erfahren, hat Ihr CPC zwei hervorragende Funktionen: XPOS und YPOS. Bei PRINT XPOS, YPOS wird Ihnen die aktuelle Position des Cursors ausgegeben.

Noch einmal zurück zum Begriff relativ, um die wirkliche Position zu errechnen brauchen Sie nur die X und Y-Werte in Ihrem Befehl zu den X und Y-Werten der Cursorposition hinzu addieren und erhalten so die absolute Position, die Sie erreichen wollen.

2.7 WINDOW-TECHNOLOGIE

In der letzten Zeit ist das Wort WINDOW genauso ein Bewertungspunkt geworden, wie die in Kapitel 2.1 erwähnte hohe Auflösung.

Was sind nun Windows eigentlich? Die deutsche Entsprechung für Window ist Fenster, dieser Begriff deutet schon ungefähr an, was Windows sind. Ein Window ist ein fester Bereich auf dem Bildschirm Ihres Computers, den Sie wie einen eigenständigen Bildschirm behandeln können. Computer, die die Windowtechnologie sinnvoll einsetzen, geben dem Benutzer die Möglichkeit, mehrere Windows zu benutzen. Dieses ist bei Ihrem CPC der Fall.

Wofür ist aber nun die Windowtechnologie gut? Hier kommt wieder ein anderes Lieblingswort der Informatik ins Spiel, die Ergonomie. Früher wurde dieser Begriff nur auf die Hardware (also auf Computer, Bildschirm, Drucker, Speichermedieen ...) bezogen. Wenn ein System einen guten Ergonomiegrad aufweist, so bedeutet das, daß seine Gestaltung, also zum Beispiel, Höhe der Tastatur, Winkel des Bildschirms, Geräuschpegel des Druckers ..., an die Bedürfnisse des Menschen so angepaßt ist, daß längeres Arbeiten mit dem System keine allzu große körperliche Belastung bedeutet.

In jüngster Zeit wurde der Ergonomiebegriff auch auf die Software (also auf die Computerprogramme) ausgedehnt. Die Wissenschaftler fanden heraus, daß die optische Gestaltung eines Programms, die Darbietung der Daten, die Antwortzeiten des Systems und ähnliches entscheidenden Einfluß auf das ermüdungsfreie Arbeiten haben. Außerdem kann man mit einem graphisch gut angepaßten Programm viel effektiver und schneller arbeiten, als mit einem schlecht angepaßten.

Warum nun Windows für einem Computer, der

Homecomputerbereich anzusiedeln ist, wie Ihr CPC 464? Hier greift nun der letzte Punkt, die Effektivität un Schnelligkeit. Die beiden meistgenutzten Computerthemen (Spiele und kommerzielle Programme) profitieren sehr davon.

Windows werden sehr oft bei kommerziellen Programmen eingesetzt. Wir werden dieses in der Dateiverwaltung des Kapitels 7.2. aufzeigen. Hier werden Windows verwendet, um Informationen, die der Benutzer ständig haben muß (Aktueller Datensatz und Feldnummer), im oberen Bildschirmteil angezeigt. Die Information wird in diesen Windows nicht verändert. Sie können aber auch mit dem Befehl ORIGIN ein Graphikwindow definieren. Bei einer von Ihnen geschriebenen Tabellenkalkulation werden die Daten einerseits tabellarisch angezeigt, und andererseits in dem definierten Grafikwindows in Form eines Balken- oder Kuchendiagramms wiedergegeben.

Der zweite Bereich, in dem Windows sehr nützlich sein können, ist der Bereich der Spiele. Reservieren Sie einen Bereich für eine Straße, die Sie immer am unteren Bildschirmrand gezeichnet haben möchten, mittels eines Windowbefehles. Sie brauchen dann diese Straße nicht immer wieder neu zu zeichnen, da solche Dinge, werden Sie von Basic aus programmiert, immer sehr zeitkritisch sind, und Ihr ganzes Spiel bis zur Langweiligkeit verlangsamen können.

Sie sollten mit dem Befehl WINDOW experimentieren. Eine Möglichkeit haben Sie in der Dateiverwaltung, natürlich können Sie auch einige kleine Programme selbst schreiben. Im Handbuch sind ja schon Beispielprogramme abgedruckt. Wie wäre es zum Beispiel mit 8 Windows, in denen 8 verschiedene Graphiken stehen. Oder bewegen Sie ein Window über den Bildschirm, indem Sie die Eckwerte verändern. Es gibt viele Einsatzmöglichkeiten für die Windows.
Um Ihnen ein paar Anregungen beim Experimentieren mit den Windows zu geben, haben wir nachfolgend zwei einfache Programme erstellt. Machen Sie auch ruhig solche Programme,

selbst wenn Sie nicht viel Sinn geben sollten. Wichtig ist nur, daß Sie den Umgang mit Windows lernen, um diese später optimal in Ihren Programmen einsetzen zu können.

1. Nachfolgendes Programm erzeugt 8 Windows und schreibt diese mit Zeichen voll, die Sie aber nicht erkennen können, da die Hintergrundfarbe verändert wurde.
Sie sollten die Werte in den Zeilen 500-510 verändern. Dieses sind die Werte für die Windows. Die erste Reihe ist die X-Richtung (Werte zwischen 1 und 80), die zweite Reihe beinhaltet die Y-Werte (zwischen 1 und 25). Natürlich können Sie auch mehr Windows auf dem Bildschirm erstellen. Dann müssen Sie allerdings die Werte in den Schleifen von 8, auf den von Ihnen gewünschten Wert heraufsetzen.

```
10 MODE 2
20 FOR x=1 TO 8
30 READ spalte (x)
40 NEXT x
50 FOR y=1 TO 8
60 READ zeile (y)
70 NEXT y
80 PAPER 5
90 FOR x=1 TO 8
100 FOR y=1 TO 8
110 WINDOW spalte(x),spalte(x)+5,zeile (y),zeile (y)+5
120 PRINT"*********"
130 NEXT y
140 NEXT x
500 DATA 1,80,20,60,30,45,27,50
510 DATA 9,19,12,25,1,3,18,1
```

Erklärung des Programms:

10	Einstellen des 80-Zeichen Modus.
20-40	Schleife zum Einlesen der Werte X-Richtung.
50-70	Schleife zum Einlesen der Werte Y-Richtung.
80	Einstellen der Hintergrundfarbe.
90-100	Eröffnung der Schleife für X/Y.
110	Erstellen des Windows.
120	Ausgabe einer Zeichenkette.
130-140	Ende der Schleifen.
500-510	DATA-Zeilen

2. Das zweite Programm schreibt den Bildschirm mit einem Zeichen voll und löscht dann jeweils ein Drittel auf Tastendruck. Eine solche Routine läßt sich gut in kommerziell orientierte Programme einbauen, da es dort oft nötig ist, ganze Bildschirmteile zu löschen und dieses mit Hilfe eines WINDOWS sehr schnell erledigt werden kann.

```
10 MODE 2
20 WINDOW #2,1,80,1,8
30 WINDOW #3,1,80,9,17
40 WINDOW #4,1,80,17,24
50 FOR n=2 TO 4
60 FOR mn=1 TO 639
70 PRINT"*";
80 NEXT
90 NEXT
100 FOR x=2 TO 4
110 WINDOW SWAP x
120 a$=INKEY$:IF a$="" THEN 120
130 CLS
140 NEXT
150 MODE 2
```

Erklärung des Programms:

5	Einstellen des 80-Zeichen Modus.
10-30	Definieren von 3 Windows.
35-80	Vollschreiben des Bildschirmes mit "*".
90	Eröffnen einer Schleife zum Wechseln der Windows.
100	Umstellen des Windows.
110	Warten auf Tastendruck.
120	Window löschen
130	Ende der Schleife, es wird auf das nächste Window geschaltet.
140	Zurücksetzen der 3 Windows.

2.8 SELBSTDEFINIERTE ZEICHEN

Die letzte hervorragende Eigenschaft, die wir im Bereich Graphik erwähnen möchten, sind die sogenannten selbstdefinierten Zeichen.

Was sind das nun für Zeichen? Im Handbuch ist dieses wichtige Thema leider nur sehr kurz angeschnitten.
Ein Zeichen setzt sich aus gesetzten und nicht gesetzten Punkten zusammen. Das Raster, in dem die Zeichen definiert werden, heißt Matrix. Solch eine Matrix finden Sie in Bild 1. Dort ist es eine 8*8 Matrix (8 Kästchen vertikal, 8 Kästchen horizontal). Andere gebräuchliche Matrixgrößen sind 7*8, 7*9 und bei profesionellen Systemen sogar 10*10 oder 12*12. Somit liegt der CPC also im oberen Mittelfeld. Die Größe der Matrix ist deshalb interessant, da eine größere Anzahl von Punkten (bei dem richtigen Datensichtgerät) auch besser lesbare Buchstaben bedeutet. Dieses ist genau wie bei der hohen Auflösung, deren Vorteile wir in Kapitel 2.1 diskutiert haben.

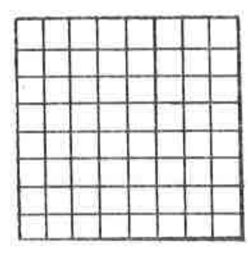




Bild 1

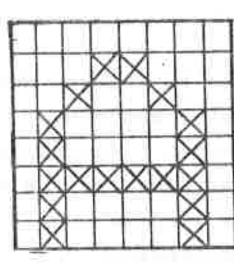


Bild 2

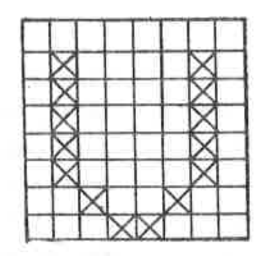


Bild 3

In Bild 2 und 3 haben wir Punkte in dieser Matrix "gesetzt". Es sind Buchstaben entstanden. In dieser Weise ist jeder Buchstabe auf Ihrem Computerbildschirm zusammengesetzt.

Der CPC hat 255 solche Zeichen, davon können Sie allerdings nur 223 Zeichen umdefinieren, das heißt, neu gestalten.

Wie geht dieses vor sich? Zuerst sollten Sie sich ein Blatt kariertes Papier nehmen und sich die 8*8 Matrix einzeichnen. Haben Sie das getan, dann probieren Sie auf dem Papier aus, welche Punkte Sie setzen möchten, und welche nicht. Dieses ist nötig, da sonst Ihre selbstdefinierten Zeichen am Ende etwas obskur aussehen könnten. Nun geht es daran, die einzelnen Punkte zu berechnen. Irgendwie muß dieses Zeichen ja nun in Ihren CPC hinein. Das ist etwas schwierig, da wir die sogenannte Binärarithmetik benötigen (diese wird im Kapitel 4 in der Maschinensprache- Einführung erläutert). Betrachten Sie bitte einmal Bild 4.

Dort finden Sie wieder eine 8*8 Matrix. Diese ist allerdings etwas erweitert worden. Über den vertikalen Spalten befinden sich nun die Zahlen 1-2-4 - 128. In den horizontalen Zeilen finden Sie noch etwas Platz, dort werden wir später noch Zahlen eintragen.
Tragen Sie nun in diese erweiterte Matrix die Punkte für Ihr Zeichen ein. Betrachten Sie nur die erste Zeile (horizontale Linie) und addieren die Zahlen, die über den gesetzten Punkten in dieser Zeile stehen. Das Ergebnis der Addition tragen Sie im dafür vorgesehenen Platz neben der Zeile ein. Gehen Sie so weiter vor mit allen 8 Zeilen. Die acht Zahlen, die Sie nun erhalten haben, sind die Werte für Ihr Zeichen. Wir demonstrieren Ihnen dieses in Bild 5 noch einmal anhand des Buchstabens E.

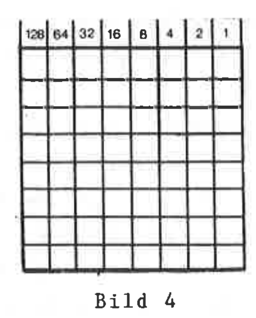


Bild 4

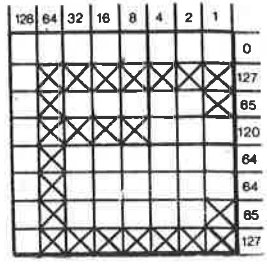


Bild 5

Wenn sie erst einmal diese Werte ermittelt haben, ist der Rest relativ einfach. Sie müssen jetzt nur noch Platz für Ihr Zeichen reservieren und dieses mittels eines Befehls einer Taste zuordnen.

Den Platz für Ihr Zeichen (oder für mehrere) reservieren Sie mit dem Befehl SYMBOL AFTER. Hierauf muß noch eine Zahl im Bereich von 32-255 folgen. Wählen Sie diesen Wert z.B. mit 240, können Sie alle Zeichen, die einen CHR$-Code größer 239 haben (also 16 Stück) umdefinieren. Die CHR$-Codes können Sie aus dem Handbuch Ihres Computers erfahren. Je kleiner Sie den Wert nehmen, desto mehr Speicherplatz verbrauchen Sie. Dieses ist allerdings beim großen Speicher des CPC nicht besonders problematisch. Außerdem hat eine möglichst kleine Zahl den Vorteil, daß Sie auch noch Platz haben, wenn Ihnen noch ein paar Zeichen einfallen, die Sie umgestalten wollen. Achten Sie bitte darauf, daß, wenn Sie einmal den Befehl SYMBOL AFTER verwendet haben, Sie ihn nicht noch einmal benutzen, da sonst die alten Zeichen wieder gelöscht werden.

Wozu nun selbstdefinierte Zeichen? Selbstdefinierte Zeichen haben den Vorteil, daß Sie sich z.B. Ihren eigenen Zeichensatz (futuristisch, technisch- mathematisch, chemisch ...) definieren können. Ganz wichtig werden Ihre selbstgestalteten Zeichen, wenn Sie Spiele machen möchten.

Jedes gute Spiel lebt von dem Aussehen der Akteure im Spiel, und es sieht nicht besonders gut aus, wenn Sie mit einem kleinen "i" große "Bs" abschießen. Dies können wir uns alle lebhaft vorstellen. Also definieren wir unser "i" einfach zu einer Kanone und unser "B" zu gefährlich aussehenden Außerirdischen um. Hier ist Ihrer Fantasie keine Grenze gesetzt.

Leider ist die Methode des Addierens von Binärzahlen eine etwas langwierige Angelegenheit. Darum stellen wir Ihnen auf den nächsten Seiten ein wirkliches Bonbon vor.

2.8.1. KOMFORTABLER ZEICHENGENERATOR

Mit dem folgenden Programm haben Sie die Möglichkeit, mit den Cursortasten innerhalb einer 8*8 Matrix Ihre eigenen Zeichen herzustellen.

Nach Eingeben und Starten des Programms erscheint ein Rechteck und in der linken oberen Ecke ein kleiner Punkt. Dieses ist in diesem Programm Ihr "Cursor". Mit den Cursortasten läßt sich dieser "Cursor" innerhalb des Rechtecks hin- und herbewegen. Wollen Sie nun einen Punkt setzen, so betätigen Sie die "COPY"-Taste und ein "*" wird an der Stelle, an welcher der Cursor steht, gesetzt. Wollen Sie einen Punkt löschen, dann müssen Sie mit dem Cursor links neben das Sternchen, welches Sie löschen wollen, fahren. Dann drücken Sie die Spacetaste. Will man danach die fertige Zeichnung in Codes umwandeln lassen, drückt man die Taste "E" und nach einer kurzen Rechenzeit werden die acht Codes der Zeichnung am unteren Bildschirmrand ausgedruckt. Danach wird Ihnen die Frage gestellt, welches Zeichen Sie umdefinieren möchten. Drücken Sie die Taste, mit der Ihr Zeichen aufgerufen werden soll und "ENTER". Es ist sinnvoll, daß Sie sich die Codes Ihres neuen Zeichen notieren, damit Sie später noch wissen, welche Taste mit welchem Zeichen belegt ist. Als letztes folgt beim Programmablauf die Frage, ob Sie noch weitere Zeichen definieren möchten. Diese beantworten Sie entweder mit "J" für Ja oder mit "N" für Nein, je nachdem ob Sie weitere Zeichen definieren wollen oder nicht.

Zu beachten ist, daß bei einem Start mit "RUN" alle vorher definierten Zeichen gelöscht sind. Wollen Sie Ihre Zeichen erhalten, so starten Sie das Programm mit "GOTO 40".

```
10 SYMBOL AFTER 32
20 IF y=0 THEN y=1
30 DIM b$(8): DIM c$(8):DIM c(8)
40 MODE 1
50 GOTO 240
60 x=1:y=1:z=0
70 LOCATE x,y
80 PLOT x*16,(128-y*16)+8,1
90 a$=INKEY$:IF a$="" THEN 90
100 PLOT x*16,(128-y*16)+8,0
110 z=0
120 y=y+(1 AND a$=CHR$(241))-(1 AND a$=CHR$(240))
130 IF y>8 THEN y=8
140 IF y=0 THEN y=1
150 x=x+(1 AND a$=CHR$(243))-(1 AND a$=CHR$(242))
160 IF x>8 THEN x=8
170 IF x=0 THEN x=1
180 IF a$=" "THEN z=2
190 IF a$=CHR$(224)THEN z=1
200 IF z=1 THEN PRINT"*"
210 IF z=2 THEN PRINT" "
220 IF a$="e" GOTO 320
230 GOTO 70
240 WINDOW#0, 10,17,10,18
250 ORIGIN 143,127
260 PLOT 0,0,1
270 DRAW 0,130
280 DRAW 132,130
290 DRAW 132,0
300 DRAW 0,0
310 GOTO 60
320 b$=""
330 PLOT 0,0
340 MOVER 10,120
350 FOR m=1 TO 8
360 FOR n=1 TO 8
370 IF TESTR(0,0)=1 THEN T$="1" ELSE T$="0"
380 b$(m)=b$(m)+T$
390 MOVER 16,0
400 NEXT
```

```
410 MOVE 10,120-(m*16)
420 NEXT
430 FOR x=1 TO 8:c$(x)="&X"+b$(x)
440 c(x)=VAL(c$(x)):NEXT
450 WINDOW #1,1,39,20,25
460 WINDOW SWAP 0,1
470 PRINT c(1);c(2);c(3);c(4);c(5);c(6);c(7);c(8)
480 INPUT"WELCHES ZEICHEN SOLL UMDEFINIERT WERDEN?",a$
490 a=ASC(a$)
500 SYMBOL a,c(1),c(2),c(3),c(4),c(5),c(6),c(7),c(8)
510 PRINT"SOLLEN WEITERE ZEICHEN UMDEFINIERT WERDEN?"
520 in$=INKEY$:IF in$="" THEN 520
530 WINDOW SWAP 1,0
540 IF in$="j"THEN RUN
550 MODE 1
560 PRINT"PROGRAMMENDE"
```

Erklärung zum Listing

In den Zeilen 10-50 ist die sogenannte Initialisierung, daß heißt, dort werden alle Vorbereitungen zur Zeichendefinition getroffen. Dann erfolgt ein Sprung in eine Unterroutine, in der die 8*8 Matrix gezeichnet wird. Das Fenster (Window) ist deshalb eine Zeile größer gewählt, damit die Zeichnung nicht nach oben gescrollt (geschoben) wird, wenn ein Punkt in der unteren rechten Ecke gesetzt wird.

Nach dem Rücksprung wird der Textcursor in die linke obere Ecke gesetzt. Gleiches gilt für den Zeichencursor. In Zeile 100 erfolgt eine Tastaturabfrage, die in den Zeilen 120-220 verwertet wird. Wird die Taste "E" gedrückt (Zeile 220), erfolgt ein Sprung zur Zeichnungsauswertung. In dieser Auswertung wird zuerst in einer Reihe getestet, ob ein Punkt gesetzt ist oder nicht (Zeile 370). Ist ein Punkt gesetzt, so wird zu der Variablen B$(M) eine eins hinzugestellt, andernfalls eine null. So entsteht für jede Zeile eine achtstellige Binärzahl. In den Zeilen 430 und 440 werden die Binärzahlen in dezimale Werte umgewandelt und in den Variablen C(W) abgespeichert. Danach wird ein Textfenster aufgebaut, in dem die Codes ausgegeben und die folgenden Fragen gestellt werden. In Zeile 500 wird das Zeichen, das

Sie erstellt haben, in den Zeichensatz aufgenommen. Als letztes werden durch den Befehl "MODE 1" alle Windows aufgehoben.

Um Ihnen zu verdeutlichen, was man mit dem Zeichengeneratorprogramm herstellen kann, haben wir eine Tabelle der Codes für die deutschen Umlaute angefertigt. Gleichzeitig haben wir ein kleines Programm geschrieben, das es Ihnen erlaubt, diese Umlaute mit der Tastatur abzurufen. Hierbei werden die Tasten ":", ";" und"^" redefiniert.

Änderungstabelle:

Umlaut	Taste	Tastennummer	Codes
ä	:	58	102,0,60,6,62,102,62,0
Ä	*	42	102,0,60,102,126,102,102,0
ö	;	59	102,0,60,102,102,102,60,0
Ö	+	43	102,60,102,102,102,102,60,0
ü	^	94	102,0,102,102,102,102,62,0
Ü	SHIFT+^	163	102,0,102,102,102,102,60,0

Hier nun das Programm. Nach dem Start werden die Umlaute definiert und der Bildschirm gelöscht. Noch eine Anmerkung: Auch wenn die Tasten umdefiniert sind, bleiben ihre Funktionen doch bestehen.

```
10 FOR N=1 TO 6
20 FOR M=1 TO 9
30 READ A(M)
40 NEXT M
50 SYMBOL A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9)
60 NEXT N
70 CLS
80 DATA 58,102,0,60,6,62,102,62,0
90 DATA 42,102,0,60,102,126,102,102,0
100 DATA 59,102,0,60,102,102,102,60,0
110 DATA 43,102,60,102,102,102,102,60,0
120 DATA 94,102,0,102,102,102,102,62,0
130 DATA 163,102,0,102,102,102,102,60,0
```

Wir haben uns gedacht, daß Sie diese Umlaute sicher gut gebrauchen können, z.B. in Verbindung mit Briefen oder sonstigen Schreiben, die Sie mit Ihrem CPC erledigen möchten.

Nachfolgend liefern wir Ihnen noch ein kleines (aber ausbaufähiges) Programm, daß mit selbstdefinierten Zeichen arbeitet und die "fast Trickfilmqualität" dieser Graphikmöglichkeit zeigt.

In dem Programm prallt ein von rechts mit hoher Geschwindigkeit fahrender PKW auf einen stehenden LKW. Diese Bewegung läßt sich einfach mit einer Schleife und dem LOCATE-Befehl ausführen.

Versuchen Sie nun, dieses Programm auszubauen. Wie wäre es, wenn Sie die Strecke, die der PKW fährt, verändern oder auch den LKW bewegen? Außerdem könnten Sie eine "Zeitlupenstudie" des Aufpralles anfertigen. Hierzu müssen Sie die Bewegung des Fahrzeuges in der entscheidenen Phase verlangsamen (dies könnte durch eine Schleife, dem Aufruf der Verzögerungsroutine ab &bd19 oder durch Setzen eines Interrupts geschehen). Außerdem müßten Sie sich selbstdefinierte Zeichen erstellen, die die Verformung des PKWs zeigen. Probieren Sie es einfach mal!

```
5 MODE 1
10 SYMBOL AFTER 32
20 SYMBOL 91,1,2,4,8,255,255,255,63
30 SYMBOL 93,252,36,36,36,255,255,255,24
40 SYMBOL 33,127,127,127,127,127,79,15,7
60 SYMBOL 35,255,255,255,255,255,190,190,28
70 SYMBOL 36,255,255,255,255,255,124,0,0
80 SYMBOL 37,255,255,255,255,255,251,248,112
90 SYMBOL 38,127,127,127,127,127,127,127,127
100 SYMBOL 39,255,255,255,255,255,255,255,255
105 SYMBOL 40,255,255,255,255,255,255,255,255
110 SYMBOL 41,240,240,240,240,255,241,241,241
111 LOCATE 2,13:PRINT"&'')"
112 LOCATE 2,14:PRINT"!#$%"
120 FOR x=39 TO 6 STEP -1
130 LOCATE x,14:PRINT CHR$(91);CHR$(93)
135 LOCATE x,14:PRINT"  "
140 NEXT x
150 GOTO 150
```

Erklärung des Programms:

5	Einstellen des Modus.
0-110	Definieren der Zeichen.
111	Ausgabe des oberen Teils PKW.
112	Ausgabe des unteren Teils.
20	Eröffnung einer Schleife für die Bewegung des Wagens in x-Richtung.
130	Ausgabe des Fahrzeuges.
135	Löschen des alten Fahrzeuges.
140	Ende der Schleife, der Wagen wird nach links bewegt.

2.9 GRAPHIKZEICHEN

Ihr CPC hat noch eine weitere sehr positive Eigenschaft. Damit diese nicht "verkümmert", möchten wir sie etwas näher betrachten. Es ist die Möglichkeit der Graphikzeichen.

Sie werden sich sicherlich fragen, warum wir Angst haben, daß die Graphikzeichen zu kurz kämen. Nun, leider ist ihre Benutzung nicht ganz einfach, und viele Computerbenutzer sind auch nach einem Jahr noch nicht in der Lage, mit den Graphikzeichen Ihres Computers umzugehen, sofern er diese Fähigkeit besitzt. Zu diesen sollen Sie nicht gehören!

Warum nun Graphikzeichen? Wir sprachen im vorigen Teilkapitel die selbstdefinierten Zeichen an. Etwas ähnliches sind auch die Graphikzeichen Ihres CPC, nur, daß Sie diese nicht mehr "zu Fuß" definieren müssen, sondern der Computer diese für Sie bereithält.

Sie können die Zeichen, sowie deren Codes aus dem Anhang Ihres Handbuches erfahren. Dort sind alle Zeichen abgebildet, die Ihrem Computer zur Verfügung stehen. Hierbei befinden sich dann auch die Graphikzeichen, wie Männchen, eine Bombe u.s.w.

Nun, wie "erreichen" Sie diese Zeichen? Sie müssen hier den Weg über die Funktion CHR$ (Argument) gehen. Möchten Sie also zum Beispiel eine Bombe auf dem Bildschirm errscheinen lassen, so geben Sie ein: PRINT CHR$(252).

So können Sie mit jedem Graphikzeichen verfahren. Natürlich können Sie diese Zeichen bewegen, beispielsweise mit Hilfe von LOCATE und Print. Probieren Sie doch bitte einmal folgendes:

```
MODE 2
FOR Y=1 TO 25
LOCATE40,Y:PRINT CHR$(252)
LOCATE 40,Y:PRINT " "
NEXT X
```

Dieses kurze Programm läßt die Bombe vom oberen zum unteren Bildschirmrand fallen. Das geschieht mittels eine Schleife, sowie des LOCATE und PRINT Befehles.

Mit etwas Phantasie kann man sich vorstellen, welch gute Spiele sich so programmieren lassen, mit etwas Geschick, kann dies bis zur Trickfilmqualität gehen.

Nachfolgend finden Sie ein weiteres Beispielprogramm, welches vor allem die Schnelligkeit dieser Methode zeigt. Das Programm endet etwas seltsam mit einer Fehlermeldung, aber das ist nicht weiter schlimm. Vielleicht finden Sie ja heraus, warum. Ein Tip: Lassen Sie sich nach Abbruch einmal den Wert für Y ausgeben. Der ist dann der Schlüssel zu Ihrer Fehlersuche.

```
10 MODE 2
20 FOR x=1 TO 25
30 t=36
40 q=12
50 LOCATE t,x:PRINT STRING$(&8,CHR$(249))
60 NEXT x
70 y=1
80 FOR x=q TO q+4
90 LOCATE y,x:PRINT STRING$(&8,CHR$(250))
100 NEXT x
110 FOR x=q TO q+4
120 LOCATE y,x:PRINT STRING$(&8," ")
130 NEXT x
140 y=y+8
150 GOTO 80
```

Erklärung des Programmes:

10	Einstellen des 80-Zeichen Modus.
20	Eröffung der Schleife für die Abwärtsbewegung.
30	Anfangswert für Abwärtsbewegung der Männchen.
40	Y-Wert für die feststehenden Männchen.
50-60	Schleife zur Ausgabe der vertikalen Männchen.
70	Y wird auf 1 gesetzt.
80-100	Schleife für die horizontale Bewegung in 4er Schritten.
110-130	Schleife zum Löschen des letzten Männchen-Blockes.
140	Erhöhung des Wertes, damit die Männchen sich in Blöcken vorwärtsbewegen.

2.10 DER BILDSCHIRMSPEICHER

Der Hauptspeicher Ihres CPC hat eine Größe von 64k. Das sind 65536 Bytes. Für Ihre Basicprogramme stehen Ihnen davon ca. 43k frei zur Verfügung.
Der komplette Speicher ist in 4 Blöcke zu je 16k aufgeteilt. Diese Blöcke heißen Banks und sind von 0 bis 3 durchnummeriert. Bank 0 repräsentiert hier den Bereich von der Speicheradresse 0(dec) bis 16383(dec), Bank 3 den Bereich von 49152(dec) bis 65535(dec). Wenn Sie Ihren Computer einschalten, so belegt der Bildschirmspeicher die Bank 3. Unabhängig davon, in welchem der drei Modi Sie sich befinden, werden also grundsätzlich 16k für den Bildschirmspeicher reserviert. Geben Sie nun bitte die folgenden 4 Zeilen ein und starten Sie das Programm mit "RUN".

```
10 REM Bildschirm 1
20 MODE 2 : PRINT "a"
30 FOR t = &C000 to &FFFF STEP &800
40 PRINT BIN$(PEEK(t)),8) : NEXT t
```

Wenn Sie alles richtig gemacht haben, dann erhalten Sie folgende Informationen :

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0
0 0 0 0 1 1 0 0
0 1 1 1 1 1 0 0
1 1 0 0 1 1 0 0
0 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0
```

Deutlich können Sie das Bitmuster erkennen, aus denen das kleine "a" besteht, welches wir in der Zeile 10 an die linke obere Bildschirmecke gebracht haben.

Ändern wir das kleine Programm wie folgt ab.

```
10 REM Bildschirm 2
20 FOR t = &C000 to &FFFF STEP &800
30 FOR a = 0 TO 3
40 PRINT BIN$(PEEK(t+a),8);" ";
50 NEXT a : PRINT
60 NEXT t
```

Geben Sie im Direktmodus jetzt "MODE 2" und anschließend "RUN" ein. Sie erhalten vier Bitmusterblöcke, auf denen wir ganz deutlich die Buchstaben R e a d erkennen können. Das ist nicht verwunderlich, da in der ersten Zeile das Wort "Ready" steht, welches durch das MODE x Kommando erzeugt wurde.
Was aber geschieht, wenn wir einen anderen Modus wählen? Da bekanntlich Probieren über Studieren geht, geben Sie "MODE 1" und erneut "RUN" ein. Das Ergebnis fällt nicht wie erhofft aus. Wir können zwar immer noch ein "R" und ein "e" erkennen, aber an der Anzahl der gesetzten Bits hat sich im Vergleich zu "MODE 2" nichts geändert. Jeder Buchstabe ist auf 2 Bitmusterblöcke aufgeteilt. Aber wider Erwarten stehen alle rechten 4 Bits eines jeden Blockes auf "0". Wieso erscheint der Buchstabe doppelt so groß, wenn doch nicht mehr Bits gesetzt sind, als im 80 Zeichen Modus? Ein weiterer Versuch soll uns helfen, diesem Phänomen auf die Schliche zu kommen.

Geben Sie bitte ein : PEN 2

```
MODE 1
RUN
```

Wieder lassen sich die beiden Anfangsbuchstaben der Ready Meldung in der ersten Zeile erkennen. Auch hier ist jeder Buchstabe auf je 2 Bitmusterblöcke aufgeteilt. Der Unterschied besteht darin, daß diesmal jeweils die linke Hälfte eines Blockes auf Null gesetzt ist und die markanten

Bits, die unseren Buchstaben darstellen auf der rechten Seite zu finden sind.
Die Erklärung zu diesen Fragen ist im Aufbau des Bildschirmspeichers in Abhängigkeit vom jeweils angewählten MODE zu suchen. Der 16k Bildschirmbereich ist für 2 Funktionen zuständig. Zum einen wird hier das Bitmuster des dargestellten Zeichens fixiert, andererseits sind die gleichen Bits auch noch für die Farbe, in der das Zeichen dargestellt werden soll, verantwortlich. Spätestens an diesem Punkt ist es erforderlich, zwischen den einzelnen Modi zu differenzieren.

2.10.1 MODE 2

Im Mode 2 ist die Angelegenheit relativ einfach. Jedes gesetzte Bit entspricht hier einem gesetzten Pixle (Bildpunkt), der dann in der Zeichenfarbe ausgegeben wird. Eine "1" bedeutet dabei die Zeichenfarbe 24 (hellgelb), eine "0" im Bitmuster kennzeichnet die Farbe 1 (blau). Um Ihnen zu verdeutlichen, daß auch "Blau" eine Zeichenfarbe ist, geben Sie nun bitte die folgenden Zeilen im Direktmodus ein:

```
MODE 2
PAPER 1 : PEN 0
CLS
```

Der Bildschirm ist nun gelb geworden, die Zeichenfarbe hat zu blau gewechselt. Starten Sie das Programm "Bildschirm 2" erneut mit "RUN". Jeder gesetzte Bildpunkt wird nun durch eine "0" im Bitmusterblock gekennzeichnet. Die Farbe Blau ist somit zu der neuen Zeichenfarbe geworden.
Fassen wir zusammen : Im Mode 2 beinhaltet jedes Bit im Bildschirmspeicherbereich die Information, mit welcher Farbe der entsprechende Bildpunkt gezeichnet werden soll.

0 = Blau
1 = Gelb

2.10.2 MODE 1

Schalten Sie nun Ihren CPC in den MODE 1. In dieser Betriebsart mit je 40 Zeichen/Zeile stehen Ihnen vier Zeichenfarben zur Verfügung. In jedem Byte des Bildschirmspeichers wird die Information verankert, welcher Bildpunkt welche Farbe haben soll.
Um Ihnen zu verdeutlichen, wie Ihr CPC die Farbe und den zu setzenden Bildpunkt speichert, müssen wir einmal ein einzelnes Byte betrachten.

```
          ----------------------------------
Bit Nr.   I 7 I 6 I 5 I 4 II 3 I 2 I 1 I 0 I
          ----------------------------------
```

Im Modus 1 wird jedes Byte in 2 Tetraden zu je 4 Bits aufgeteilt, die hier duch den doppelten Trennstrich gekennzeichnet sind. Die zwei Bits mit der gleichen Position in den beiden Tetraden werden zu einem Paar zusammengefaßt. Ein Paar bilden immer die Bits (3,7), (2,6), (1,5)und (0,4). Mit diesen vier Paaren müssen nun alle acht vorhandenen Bildpunkte angesprochen werden. Da dieses offensichtlich unmöglich ist, faßt man je zwei benachbarte Bildpunkte zusammen und ordnet ihnen eines der vier genannten Bitpaare zu. Auf diese Weise ist es möglich, jedem Pixlepaar auf dem Bildschirm eine von 4 möglichen Farben zuzuordnen.
Dabei beinhalten die Bits 3,7 die Information für das ganz linke Pixlepaar.
In der nachfolgenden Tabelle sehen Sie, welche Farben die Zustände der Informationsbits repräsentieren.

Muster der Informationsbits	Farbe
0 0............	Blau
0 1............	Hellgelb
1 0............	helles Blaugrün
1 1............	Hellrot

Lassen Sie uns das soeben erworbene Wissen in der Praxis erproben. Erinnern Sie sich noch an das merkwürdige Aussehen der Bitmusterblöcke, als wir diese im MODE 1 ausgelesen hatten? Falls Sie sich nicht entsinnen können, dann blättern Sie bitte zu Programm "Bildschirm 2" zurück und probieren die beiden Beispiele erneut aus. Betrachten wir die 1. Reihe des 1. Bitmusterblocks :

1 1 1 1 0 0 0 0

Fassen Sie die Informationsbitpaare zusammen und schauen in der Tabelle nach, welche Farbe sie anzeigen.

Bitpaar :	3,7	2,6	1,5	0,4
Wert :	0 1	0 1	0 1	0 1
Farbe :	Gelb	Gelb	Gelb	Gelb

Nehmen Sie sich bitte ein Blatt Rechenpapier zur Hand, auf welchem Sie sich 4 Quadrate mit der Kantenlänge 8 mal 8 Kästchen nebeneinander einzeichnen. Jedes Quadrat unterteilen Sie dann noch einmal in 4 senkrechte Spalten (je 2 Kästchen breit) und 8 Reihen. Jedes Kästchenpaar auf dem Papier entspricht zwei benachbarten Bildschirmpunkten. Füllen Sie dann jedes Kästchenpaar aus, wenn das entsprechende Informationsbitpaar angibt, daß hier die Farbe Gelb gesetzt ist. Lassen Sie die Kästchen leer, wenn die Informationsbits auf Blau deuten. Sollten Sie in unserem Beispiel einen anderen Wert als 00=(Blau) oder 01=(Gelb) erhalten, so haben Sie einen Fehler gemacht. Auf diese Weise entschlüsseln Sie dann jede Zeile der auf dem Bildschirm angebotenen Bitmusterblöcke. Als Ergebis erhalten Sie ein breites "R" und ein breites "e" auf dem Papier.
Wechseln Sie nun die Zeichenfarbe mit PEN 2. Geben Sie ein : CLS und RUN. In gewohnter Weise fassen wir wieder die Informsationsbitpaare zusammen und stellen fest, daß alle den Wert "1 0" haben. Ein Blick auf die Farbtabelle bestätigt die Vermutung, daß diese Kombination die Farbe

"helles Blaugrün" erzeugt.
Mit diesem Wissen im Hinterkopf wird es Ihnen sicherlich nicht mehr schwerfallen, eigene Vierfarbzeichen zu entwerfen. Beachten Sie dabei aber, daß jedes Zeichen nur die halbe Breite eines normalen Schriftzeichens im MODE 1 hat. Ferner ist es auch nicht möglich, diese Zeichen mit SYMBOL oder SYMBOL AFTER in die Zeichentabelle zu übernehmen. Vierfarbzeichen und Multicolorzeichen, auf die die Sprache im nächsten Abschnitt kommen wird, lassen sich nur in den Bildschirm "poken". Sie müssen vorher in DATA Zeilen abgelegt werden, aus denen sie dann während des Programmlaufs ausgelesen werden.
Die Formel zur genauen Berechnung der Bildschirmposition, an die ein Vielfarbzeichen gepoked werden soll, finden Sie am Ende dieses Abschnitts.

2.10.3 MODE 0

Dieser Betriebsmodi ist signifikant für die Leistungsfähigkeit Ihres CPC, viele Farben auf kleinstem Raum darstellen zu können. Bedauerlicherweise geht aber die mögliche Farbenvielfalt zu Lasten der Einzelpunktauflösung. Bedenken Sie, daß, wenn Sie jedes Pixle in 16 Farben ansteuern wollten, Sie einen Bildschirmspeicher von 64k benötigen würden. Wie auch schon im Mode 1, so greift man in dieser Betriebsart wieder zu dem Trick, daß man mehrere benachbarte Bildpunkte mit einem Farbwert ansteuert. Waren es im Vierfarbmodus 2 benachbarte Bildpunkte, die von einer Farbe gemeinsam angesprochen wurden, so sind es im Multicolormodus 4 Pixles. Mit anderen Worten kann man sagen, daß jedes Byte des Bildschirmspeichers genau 4 benachbarte Bildpunkte in ihrer Farbe steuert. Aus der nachfolgenden Tabelle können Sie ersehen, welche Farben Ihnen zur Verfügung stehen.

Farbtabelle für Multicolormodus

Bin	Hex	Dez	Farbe
0000	0	00	Blau
0001	1	01	Hellgelb
0010	2	02	helles Blaugrün
0011	3	03	Hellrot
0100	4	04	Leuchtendweiß
0101	5	05	Schwarz
0110	6	06	Hellblau
0111	7	07	helles Magenta
1000	8	08	Blaugrün
1001	9	09	Gelb
1010	A	10	Pastellblau
1011	B	11	Rosa
1100	C	12	Hellgrün
1101	D	13	Pastellgrün
1110	E	14	Blau/Gelb blinkend
1111	F	15	Rosa/Himmelblau blinkend

Mit einem kleinen Programm wollen wir versuchen, diese 16 Farben auf den Bildschirm zu zaubern.

```
10 REM Farbenbeispiel
20 DATA &00,&11,&22,&33,&44,&55,&66,&77,
        &88,&99,&AA,&BB,&CC,&DD,&EE,&FF
30 MODE 0
40 FOR d = 0 TO 31 STEP 2
50 READ x
60 FOR t = &C000 to &FFFF STEP &800
70 POKE t+d,x
80 NEXT t : NEXT d
90 GOTO 90
```

Das Kommando "RUN" startet unser kleines Programm. Erstaunlicherweise bekommen Sie aber nicht die erhofften 16 Farben zu sehen, sondern ein kunterbuntes Muster. Was haben wir falsch gemacht? Wie schon im Mode 1, so sind auch hier die Farbwerte, die in einem Byte verschlüsselt sind, nicht in 4 zusammenhängenden Bits zu suchen. Vielmehr ist der Code über das ganze Byte verstreut. Die Information für die rechten 4 Bildpunkte wird in den Bits 1,3,5,7 versteckt, der linke Farbwert in den Bits 0,2,4,6. Damit aber noch nicht genug. Die Reihenfolge, in der diese Farbinformationsbits zusammengehören, entspricht keinesfalls der Reihenfolge, die Sie soeben gelesen haben. Hier die richtige Reihenfolge :

```
links  liegende Bildpunkte = 1,5,3,7
rechts liegende Bildpunkte = 0,4,2,6
```

Mit diesem Wissen ist es uns jetzt auch möglich, alle vorhandenen Farben anzusprechen.

Bit Nr. 7 6 5 4 3 2 1 0		links	rechts	Byte
0 0 0 0 0 0 0 0	=	00	00	0
1 1 0 0 0 0 0 0	=	01	01	192
0 0 0 0 1 1 0 0	=	02	02	12
1 1 0 0 1 1 0 0	=	03	03	204
0 0 1 1 0 0 0 0	=	04	04	48
1 1 1 1 0 0 0 0	=	05	05	240
0 0 1 1 1 1 0 0	=	06	06	60
1 1 1 1 1 1 0 0	=	07	07	252
0 0 0 0 0 0 1 1	=	08	08	3
1 1 0 0 0 0 1 1	=	09	09	195
0 0 0 0 1 1 1 1	=	10	10	15
1 1 0 0 1 1 1 1	=	11	11	207
0 0 1 1 0 0 1 1	=	12	12	51
1 1 1 1 0 0 1 1	=	13	13	243
0 0 1 1 1 1 1 1	=	14	14	63
1 1 1 1 1 1 1 1	=	15	15	255

Ändern Sie das Programm "Farbenbeispiel" derart ab, daß Sie die unter Byte vorgegebenen Werte in die DATA Zeile eintragen. Die Zahlen unter Byte sind Dezimalzahlen, so daß Sie kein "&" vorstellen müssen. Hier die neue DATA Zeile :

```
20 DATA 0,192,12,204,48,240,60,252,3,
        195,15,207,51,243,63,255
```

Wenn Sie das so modifizierte Programm starten, dann erhalten Sie die 16 Farben in der richtigen Reihenfolge. Genau genommen werden Sie nur 15 Balken zählen. Das liegt darin begründet, daß der 1. Balken in der Farbe 0, also der Hintergrundfarbe gezeichnet ist.

2.10.4 POSITION EINES ZEICHENS

In den letzten Abschnitten war oft die Rede von Multicolorzeichen, ihrem Aufbau und wie man solche Zeichen selbst berechnen kann. In diesem Abschnitt bekommen Sie ein kleines Hilfsprogramm vorgestellt, mit dem Sie diese Farbmuster auf dem Bildschirm positionieren können.

```
 10 REM Position eines Zeichens
 20 MODE 0
 30 INPUT "Welche Spalte";s
 40 LOCATE 1,1
 50 INPUT "Welche Zeile ";z
 60 LOCATE 1,1
 70 INPUT "Farbmuster   ";farbe
 80 LOCATE 1,1
 90 basis = &C000
100 FOR adresse = basis TO basis + &3FFF STEP &800
110 POKE adresse + (s-1) + ((z-1)*80),farbe
120 NEXT adresse
130 GOTO 30
```

Diese Programm poked kleine Rechtecke in der Größe des Cursors im Mode 2 an die von Ihnen durch Zeile und Spalte definierte Position. Die Frage nach dem Farbmuster ist nicht identisch mit den Farbtabellen Ihres CPC Handbuches. Hier müssen Sie einen Wert eintragen, den Sie selbst errechnet haben. Das sollte Ihnen aber keine Probleme bereiten, wenn Sie das Kapitel 2.10.3 aufmerksam durchgelesen haben. In Zeile 90 des Programms wird der Variablen "basis" ein Wert zugewiesen. Dabei handelt es sich um die Standardadresse, bei der der Bildschirmspeicher beginnt. Im Kapitel 6 dieses Buches werden Sie eine Maschinenroutine finden, mit der man den Beginn des Bildschirmspeichers umlegen kann. Entsprechend dem veränderten Bildschirmspeicherbereich muß dann natürlich auch die Basisadresse angepaßt werden. Schrittweite und Endadresse brauchen Sie nicht zu beachten,

da sie sich automatisch einstellen. Die Zeile 110 ist das Kernstück. Hier wird der Farbklecks an die richtige Stelle gebracht. Beachten Sie dabei, daß die Position am linken oberen Bildschirmrand (die sogenannte HOME Position) die Kordinaten 1,1 besitzt. Das Programm verfügt in der hier vorgestellten Form über keine Routine, in der die eingegebenen Werte auf Gültigkeit überprüft werden.

STEP &800

Ihnen ist sicherlich nicht verborgen geblieben, daß in allen Programmen, bei denen wir Zeichen aus dem Bildraum auslesen oder hineinpoken, immer wieder die Schrittweite "STEP &800" auftaucht. Dazu hier die entsprechenden Erklärungen. Im Gegensatz zu vielen anderen Rechnern, bei denen die acht Reihen, aus denen jedes Zeichen besteht, auch im Bildschirmspeicher acht hintereinander liegende Bytes sind, werden bei Ihrem CPC alle ersten Reihen aller 2000 Bildschirmpositionen (25 Zeilen mit je 80 Zeichen) zu einer Kette aufgereiht. In der zweiten Kette befinden sich dann alle zweiten Reihen der 2000 Schreibstellen. Die erste Adresse der ersten Kette ist identisch mit dem Beginn des Bildspeichers, hat also die Adresse &C000 oder 49152(dezimal). Die nächste Kette beginnt bei &C800 oder 51200(dezimal). Alle weiteren Werte erhält man, indem man zu dem vorausgegangenen 2048 addiert. Unser "STEP &800" ist nichts anderes, als die hexadezimale Schreibweise von 2048.

	Hex	Dezimal
1. Reihe	C000	49152
2. Reihe	C800	51200
3. Reihe	D000	53248
4. Reihe	D800	55296
5. Reihe	E000	57344
6. Reihe	E800	59392

7. Reihe F000 61440
8. Reihe F800 63488

Diese Einteilung des Bildspeichers wirft jetzt aber eine neue Frage auf. Wir haben 2000 Schreibstellen zur Verfügung, jede Kette ist aber 2048 Bytes lang. Was geschieht mit den nicht sichtbaren 48 Bytes am Ende jedes Blockes? Die Antwort ist denkbar einfach. Diese Bytes sind ungenutzt. Mit einer kleinen Einschränkung allerdings. Ändert man den Offset des Bildschirms, um beispielsweise ein horizontales Scrolling zu erzeugen, dann treten diese Bytes in Erscheinung. Seien Sie also vorsichtig, wenn Sie diesen Platz für eigene Maschinenroutinen nutzen wollen. Eine ausführliche Beschreibung des Offsets mit der damit verbundenen Möglichkeit des horizontalen Scrollens finden Sie im Kapitel 6, Abschnitt "Sprungadressen".

2.11 ALTERNATIVE ZEICHEN EINMAL ANDERS

Im Kapitel 2.8 haben Sie schon eine Möglichkeit, mit der Sie einen eigenen Zeichensatz entwerfen können, kennengelernt. Mit den Befehlen SYMBOL und SYMBOL AFTER sind Sie in der Lage, eigene Zeichen zu entwerfen und zu nutzen. Das mag auch für den Einsteiger ausreichen. Es gibt aber immer Situationen, bei denen man gerne noch eine andere Art hätte, mit der man eigene Zeichen definieren kann. Denkbar ist das dann, wenn der Zeichensatz nicht mehr ausreicht und man nur ein Zeichen ändern möchte. Weitere Einsatzgebiete sind sicherlich Programme, die voll in Maschinensprache geschrieben sind, oder wenn man nur 4 oder 5 Punkte innerhalb eines Zeichens verändern möchte. Für all die Leser, die früher oder später einmal in diese Verlegenheit kommen, ist dieser Abschnitt gedacht.
Geben Sie das folgende Programm ein :

```
10 REM Zeichensatz auslesen
20 MODE 1
30 FOR t = &AB80 TO &AB87
40 PRINT BIN$(PEEK(t),8)
50 NEXT t
```

Auf dem Bildschirm erhalten wir das Bitmuster des nach oben gerichteten Pfeils. Das ist das Zeichen mit dem Wert CHR$(240). Auf diese Weise lassen sich alle Zeichen, deren Charaktercode 240 oder größer ist, auslesen. Bei den anderen Symbolen ist das nicht so einfach. Deshalb verzichten wir an dieser Stelle auch darauf, dieses zu beschreiben. So einfach, wie wir auf eines der letzten 16 Zeichen zugreifen können, so einfach ist es auch, diese zu verändern. Versuchen Sie einmal die folgende Basic Zeile :

```
POKE &AB87,255
```

Starten Sie das noch im Speicher befindliche Programm wieder

mit "RUN". Unser nach oben gerichteter Pfeil hat jetzt einen "Fuß" bekommen, auf dem er stehen kann. Der Pfeil mit dem Fuß kann selbstverständlich auch auf dem Bildschirm ausgegeben werden.

```
PRINT CHR$(240) : PRINT
```

Der zweite Print Befehl dient dazu, daß das Wort "Ready" nicht direkt unter unserem Symbol zu stehen kommt und wir so unseren neu erzeugten Fuß besser bewundern können. Hier die Liste der Zeichen, auf die man mit dieser Methode frei zugreifen kann.

Hex		Dezimal		CHR$( )
von	bis	von	bis	
AB80	AB87	43904	43911	240
AB88	AB8F	43912	43919	241
AB90	AB97	43920	43927	242
AB98	AB9F	43928	43935	243
ABA0	ABA7	43936	43943	244
ABA8	ABAF	43944	43951	245
ABB0	ABB7	43952	43959	246
ABB8	ABBF	43960	43967	247
ABC0	ABC7	43968	43975	248
ABC8	ABCF	43976	43983	249
ABD0	ABD7	43984	43991	250
ABD8	ABDF	43992	43999	251
ABE0	ABE7	44000	44007	252
ABE8	ABEF	44008	44015	253
ABF0	ABF7	44016	44023	254
ABF8	ABFF	44024	44031	255

KAPITEL 3 SOUND

3.1 EINFÜHRUNG

In dem folgenden Kapitel werden einige wichtige Begriffe und Techniken der Geräusch- und Musikerzeugung erklärt. Wir haben diese Begriffe in der Reihenfolge ihres Auftretens einmal untereinander zusammengestellt.

---Anschluß des CPC 464 an eine Stereoanlage
---Grundlagen der Musik und des Sounds
---Erklärung der Soundbefehle
---Env (Hüllkurve der Lautstärke)
---Ent (Hüllkurve des Tones)
---Soundeditor
---Demos zur Spieleuntermalung

Die folgenden Ausführungen werden Ihnen teilweise etwas theoretisch vorkommen, jedoch haben wir versucht, das ganze Thema durch viele Beispiele weitgehendst aufzulockern. Halten Sie also Ihren CPC bereit, um die wichtigsten Programme einzugeben.

Bei eventuellen Unverständlichkeiten blättern Sie ruhig in Ihrem Handbuch und lesen Sie dann noch einmal die nicht verstandene Passage noch einmal.

Auch wenn Sie absolut unmusikalisch sind, so brauchen Sie nicht zu verzweifeln. Computermusik ist ein sehr mathematisch- physikalisches Thema. Wenn Ihnen auch das nicht liegt, so werden Sie sich mit Sicherheit an den schönen Melodien erfreuen, deren Listings wir am Ende des Kapitels abgedruckt haben.

3.2 ANSCHLUß DES CPC AN EINE STEREOANLAGE

Wie Sie sicher bemerkt haben, ist der eingebaute Lautsprecher für die Tonfähigkeiten Ihres CPC nicht besonders geeignet. Um diesem Übel Abhilfe zu schaffen, ist hier ausführlich erklärt, wie Sie ihren CPC an eine Stereoanlage anschließen können.

Wenn Sie es ausprobiert haben sollten, einen Kopfhörer an den Anschluß auf der Rückseite des CPC anzuschließen, wird Ihnen aufgefallen sein, daß dieser keinen Laut von sich gibt. Diese Tatsache hängt damit zusammen, daß das Signal am Ausgang nicht verstärkt ist, und so Ihr Kopfhörer auch nicht angesprochen wird.

Der Kopfhörer benötigt ein vorverstärktes Signal, bzw. einen Ton, um diesen weitergeben zu können. Um also einen Kopfhörer zu nutzen, ist es notwendig, einen Verstärker zwischenzuschalten. Ihre Stereoanlage beinhaltet einen solchen Verstärker, aber auch jeder Kassettenrecorder und jedes Radio.

Wie bekommen Sie aber nun das Signal vom CPC in Ihren Verstärker? Dazu gehört eine abgeschirmte Leitung, die an der Seite zum CPC hin einen Klinkenstecker mit 3.5 mm Durchmesser aufweist. An der Verstärkerseite ist ein weiterer Stecker notwendig. Dieser ist abhängig von Ihrer Mikrophoneingangsbuchse. Es muß deshalb der Mikrophoneingang sein, weil dieser am besten auf nicht vorverstärkte Signale reagiert. Sollten Sie keinen solchen Eingang an Ihrem Verstärker haben, so tut es auch ein Plattenspieler- oder Kassettenrecordereingang.

Verbinden Sie nun die beiden Geräte miteinander, vergessen Sie aber bitte nicht, vorher beide Geräte auszuschalten, da es sonst leicht zu Komplikationen kommen könnte.

Nachdem Sie beide Geräte verbunden und wieder eingeschaltet haben, müssen Sie noch den Verstärker auspegeln. Andernsfalls könnte es eine unangenehme Überraschung bezüglich der Lautstärke geben.

Das Auspegeln des Verstärkers können Sie am besten erreichen, indem Sie folgenden Befehl eingeben:

```
SOUND 7,284,32767,7
```

Dieser Befehl erzeugt den Ton für den internationalen Notenwert a, das von allen Musikern zum Stimmen ihrer Instrumente benutzt wird. Es ist die Note von der sich alle anderen Noten aus errechnen lassen.

3.3 GRUNDLAGEN DES SOUNDS

Die Musik, bzw. die Erzeugung von Geräuschen, ist eines der wichtigsten Gebiete, die ein guter Heimcomputer umfassen muß.
Bezüglich dieser Feststellung kann man den CPC 464 zu den Heimcomputern der Spitzenklasse zählen, da er mit seinen hervorragenden Soundeigenschaften jeden Vergleich mit anderen Rechnern leicht aufnehmen kann.

Die Grundlagen des Sounds sind in erster Linie die Sprache und die Tierlaute, denn ohne diese Eigenschaften der Lebewesen gäbe es keine vernünftige Anwendung der Tonerzeugung. Um Ihnen einmal zu verdeutlichen, wie leistungsfähig Ihr CPC ist, wollen wir an dieser Stelle einmal den Hörbereich des Menschen ansprechen. Ein Durchschnittserwachsener hat einen Hörbereich von circa 16 Hz bis circa 16 Khz. Ihr CPC kann hingegen Töne im Frequenzgang von 30 Hz bis 125 Khz erzeugen. Sie sehen also, daß Sie nicht alle Noten, die ihr CPC erzeugt, hören können. Wenn Sie aber einen Hund zu Hause haben sollten, und Sie geben den Befehl SOUND 1,1,100 ein, so wundern Sie sich bitte nicht, wenn er darauf ansprechen sollte, denn er kann diesen Ton noch hören. Das heißt, er hat einen größeren Hörbereich als der Mensch.

Wie Sie sicherlich wissen, ist ein Musikstück nichts anderes als eine Aneinanderreihung von Tönen. Es können durchaus auch nicht zueinanderpassende disharmonische Töne sein, jedoch ist dies im allgemeinen nicht so wohlklingend, wie ein gutes Lied.

Ein Lied kann aber auch, wenn die Noten in der richtigen Reihenfolge gespielt werden, schlecht klingen. Dies hängt dann zumeist mit dem Instrument zusammen, mit dem es gespielt wird.

Sie brauchen sich darüber keine Gedanken zu machen, denn mit dem CPC 464 haben Sie ein hervorragendes Instrument zur Hand.

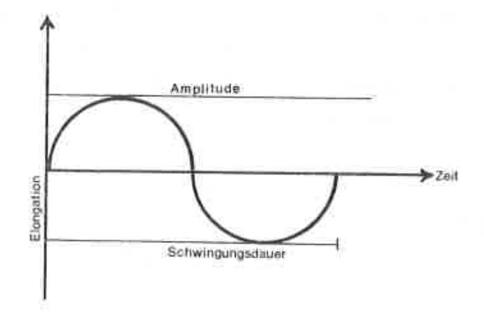
Eine der wichtigsten Grundlagen zum Verständnis der Tonerzeugung, ist die Kenntnis, wie ein Ton überhaupt aussieht.

Die heute gültige Lehre ist die, daß ein Ton sich aus Wellen, bzw. Schwingungen zusammensetzt. Folglich ist ein Ton höher als ein anderer, wenn er eine höhere Frequenz hat. Das heißt, es werden mehr Schwingungen in der gleichen Zeit ausgeführt. Die Maßeinheit für die Frequenz ist Hertz, abgekürzt mit Hz.

Hierzu ein Beispiel:

Das mittlere C hat eine Frequenz (Schwingungszahl) von 261,626 Hz.
Das internationale a hat eine Frequenz von 440 Hz, und ist somit höher, als das oben genannte c.

Um Ihnen das Aussehen eines Tones noch stärker zu verdeutlichen, haben wir eine Schwingung mit allen wichtigen Benennungen aufgezeichnet.
Diese Benennungen werden anschließend ausführlich erklärt.



Hier die versprochene Erklärung der Begriffe von links nach rechts.

1. Elongation (momentane Entfernung aus der Ruhelage)

Dieser Begriff ist für die Sounderzeugung nicht so wichtig, da er nicht sinnvoll anzuwenden ist. Dieser Wert ist lediglich die Benennung der Y-Achse.

2. Amplitude (größte Elongation oder Schwingungsweite)

Dieser Wert ist beim Sound der Wert der Lautstärke. Das heißt, je größer die Amplitude, desto lauter ist der Ton.

3. Zeit oder Tondauer

Dieser Faktor wird bei der Erstellung der Hüllkurven eine wichtige Rolle spielen. In unserer Grafik ist er die Benennung der X-Achse.

4. Schwingungsdauer

Dieser Wert ist die Zeit für eine vollständige Schwingung. Die Schwingungsdauer ist abhängig von der Frequenz. Mit steigender Frequenz, sinkt die Schwingungsdauer.

Diejenigen Leser unter Ihnen, die nicht "eingeweiht" sind in die Physik der Schwingungen, werden sich sicher fragen, wie solche Schwingungen vom CPC erzeugt werden. Dieses ist zu komplex, um es hier vollständig zu erklären, jedoch die Grundkenntnisse können wir Ihnen vermitteln.

Ein Lautsprecher, der zur Tonerzeugung im Rechner erforderlich ist, besteht einfach gesagt aus einem Magneten, einer Spule und einer Schallwand. Wird nun in einem bestimmten Rhythmus elektrischer Strom durch die Spule geschickt, so zieht sie den Magneten entweder an oder stößt

ihn ab. Stößt sie ihn ab, so schlägt der Magnet gegen die Schallwand und erzeugt eine Schwingung, die an die Luft weitergeleitet wird. In der Luft breitet sich diese Schwingung dann aus und gerät nach einem Moment auch an Ihr Ohr.

Durch den Rhythmus des elektrischen Stroms läßt sich auch die Frequenz des Tones bestimmen und somit auch seine Höhe.

Zur näheren Erklärung der Tonerzeugung schauen Sie am besten in einem Lexikon unter den oben genannten Begriffen nach. Dort werden Sie eine weiterreichende Erklärung finden.

3.4 SOUNDBEFEHLE

Die wichtigsten Soundbefehle des CPC sind:

SOUND
ENV
ENT
RELEASE

Zum ersten Befehl: SOUND ist der Grundbefehl zu Tonerzeugung des CPC. Er hat die folgenden Parameter (in ihrer Reihenfolge nach dem Befehl):

1. Kanal Status: Dieser Wert gibt dem Rechner bekannt, welcher Tonkanal benutzt wird und welche sonstigen Aktivitäten angestrengt werden sollen, wie zum Beispiel ein Rendezvous oder ein Halt.

Die einzelnen Werte für die Aktionen werden aus dem folgenden Byte errechnet:

Bit	Wert	Kommando
0	1	Ton wird zum Kanal'A geschickt.
1	2	Ton wird zum Kanal B geschickt.
2	4	Ton wird zum Kanal C geschickt.
3	8	Rendezvous mit Kanal A.
4	16	Rendezvous mit Kanal B.
5	32	Rendezvous mit Kanal C
6	64	Halt.
7	128	Rauschen.

Wollen Sie also einen Ton gleichzeitig zu allen drei Kanälen schicken, so muß der Kanalstatus den Wert 7 haben. Das heißt, daß Sie alle Werte für die gewünschten Kommandos zusammenrechnen müssen .

Mit einem Rendezvous können Sie die Tonfolge der drei Tonkanäle aufeinander abstimmen, das heißt, Sie können eine

Melodie mit Pausen spielen, ohne daß ein unangenehmes Knacken den Lautsprecher verläßt.

Wenn Sie also eine Melodie spielen wollen, und sie möchten die Töne abwechselnd aus allen drei Tonkanälen klingen lassen, so arrangieren Sie ein Rendezvous. Ein solches Rendezvous kann folgendermaßen aussehen.

```
10 READ a,b
20 IF a=-1 THEN RESTORE:GOTO 10
30 SOUND 1,a,b
40 SOUND 2,0.5*a,b
50 SOUND 4,0.25*a,b
60 GOTO 10
70 DATA 478,50,379,50,358,50,319,200,0,5,319,50,478,50,379,5
0,358,50,319,200,0,5,319,100
80 DATA 478,50,379,50,358,50,319,100,379,100,478,100,379,100
,426,200
90 DATA 0,5,426,50,379,50,0,5,379,50,426,50,478,150,0,5,478,
50
100 DATA 379,100,319,100,0,5,319,50,358,150,0,5,358,100,379,
50,358,50,319,100,379,100
110 DATA 478,100,426,100,478,200,0,5,478,50,-1,1
```

	1	2	3	4	5
Kanal A:	NO	NO	NO	RB	RB
Kanal B	NO	RC	NO	NO	RA
Kanal C	NO	RB	RA	NO	NO

In diesem Schema bedeutet NO, daß eine Note gespielt wird, und R mit dem entsprechenden Kanalbuchstaben ein Rendezvous.

Wir haben Ihnen anschließend ein Schema aufgezeichnet, aus dem Sie den zeitlichen Ablauf der vorher beschriebenen Rendezvousstruktur erkennen können. Zeitgleiche Aktionen stehen untereinander. NO bedeutet wieder eine Note, WA eine Pause.

Kanal A	NO	NO	NO	WA	NO	NO	WA	WA
Kanal B	NO	NO	NO	NO	NO	NO	WA	WA
Kanal C	NO	NO	WA	WA	WA	NO	NO	NO

Zu beachten ist, daß, wenn ein Rendezvous arrangiert wird, in dem angesprochenen Kanal eine Pause erzeugt wird. Der angesprochene Kanal ist der, der weiterspielt.
Eine weitere Möglichkeit, die Kanäle zu synchronisieren, ist es, ein Halt zu setzen. Mit diesem Halt können Sie das Ausspielen eines schon gefüllten Kanals aufhalten und mit dem Befehl RELEASE wieder aufrufen. Der Vorteil gegenüber dem Rendzvous ist es, das der Tonkanal bei einem Halt gefüllt werden kann, während das Rendezvous keine weiteren Töne in den Kanal läßt, bevor nicht das Rendezvouz ausgeführt ist.

2. Notenwert: Der zweite Wert nach dem Soundbefehl ist der Notenwert. Diesen können Sie aus dem Anhang Ihres Handbuches entnehmen. Normalerweise benötigen Sie nur die Werte der Oktave null, alle anderen Werte können Sie folgendermaßen errechnen:

Notenwert = Notenwert der Note in der Oktave 0 / 2^Oktave

Ein Beispiel dazu: Wollen Sie den Wert des C in der Oktave -2 errechnen, so rechnen Sie: 478/2^-2 oder 478/0.25

3. Tondauer: Diese ist in 100stel Sekunden anzugeben.

Ist der Wert negativ, so gibt er an, wie oft die Tondauer des ENV wiederholt werden soll. Ist er null, wird der Ton entsprechend der Tondauer des ENV gespielt.

4. Lautstärke: Sie wird von 0-7 angegeben, wenn kein ENV gesetzt ist. Ist ein ENV gesetzt, werden Werte von 0-15 angenommen.
ie beiden nachfolgenden Werte werden später noch ausführlich erklärt. Sie sind für ENV und ENT.

Der letzte Wert steht für die Geräuscherzeugung. Wird diese in allen drei Kanälen angewandt, so empfangen alle drei dasselbe Geräusch.

Der nächste Befehl, der eng mit dem Soundbefeh zusammenhängt, ist der Befehl RELEASE. Mit ihm läßt sich ein gesetztes Halt wieder rückgängig machen. Hierbei muß nur die Kanalzahl angegeben werden (A=1, B=2, C=4).
Bevor wir aber zum wichtigsten Teil, der Erklärung der Hüllkurven, kommen, wollen wir die Stimmung durch ein kleines Lied aufheitern. Es heißt:" Oh when the Saints go marching in". Tippen Sie das folgende Programm ab und speichern Sie es erst bitte auf Cassette, da wir es später weiter benutzen wollen.

3.5 DER ENV BEFEHL UND DIE LAUTSTÄRKEHÜLLKURVEN

Der ENV Befehl ist dazu gedacht, die Lautstärke eines Tons so zu verändern, so daß ein Ton entsteht, wie ihn ein Instrument spielen würde.

Es besteht die Möglichkeit, durch das Ansteigen oder Abfallen der Lautstärke während des Spielens, eine verbesserte Resonanz zu erzeugen.

Beim Einsetzen eines ENV ist darauf zu achten, daß die Lautstärke beim Soundbefehl auf Null gesetzt ist, damit die Hüllkurve voll zur Geltung kommt.

Die Hüllkurven werden sogenannte Labels zugeordnet, auch Envnummer genannt, die dann im Soundbefehl eingesetzt werden, um die Hüllkurve aufzurufen.

Dieser Label ist der erste Wert des ENV Befehls.
Ihm folgen drei Werte, die für die Schrittzahl, Schrittweite und Pausezeit stehen. Diese drei Werte können fünfmal hintereinander eingesetzt werden.

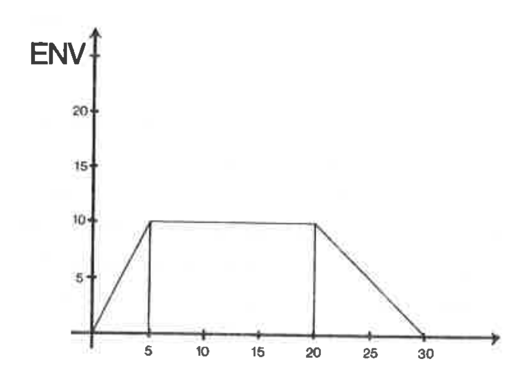
Die Schrittzahl gibt die Anzahl der Schritte an, in der die Lautstärke ansteigen bzw. abfallen soll. Die Schrittweite ergibt, multipliziert mit der Pausezeit, die Tondauer. Ist dieser Wert negativ, so fällt die Lautstärke. Die Pausezeit ist die Zeit, in der die aktuelle Lautstärke gehalten wird. Sie wird in 1/100 Sekunden gerechnet. Diese Pausezeit kann auch als Schrittdauer bezeichnet werden.

Versuchen Sie einmal in Verbindung mit unserem Lied den folgenden ENV:

```
ENV 1,5,2,1,1,0,16,5,-3,2
```

Sie werden verblüfft sein, welches Ergebnis sich ergibt. Wir wollen nun erklären, wie wir auf die oben verwendeten Werte gekommen sind.

In Ihrem Handbuch ist ein Blatt enthalten, auf dem Sie die Kurve für die Hüllkurven aufzeichnen können. Achten Sie darauf, sich für jeden Teil der Kurve die drei Werte zu notieren. Wir haben Ihnen eine solche Kurve aufgezeichnet und mit allen Werten versehen, damit Sie die Entwicklungsweise eines ENV auch völlig verstehen.



3.6 ENT UND DIE TONHÜLLKURVE

Die Tonhüllkurve ist im Prinzip dasselbe, wie die Lautstärkehüllkurve, jedoch sind die Möglichkeiten nicht so groß, wie bei der Lautstärkehüllkurve, denn mit der Tonhüllkurve ist nur ein geringer Grad an Veränderungen zu erreichen. Eine solche Veränderung ist ein Vibrato, also eine minimale Änderung der Tonhöhe, die ein Gefühl vermittelt, als wenn der Ton "zittern" würde.

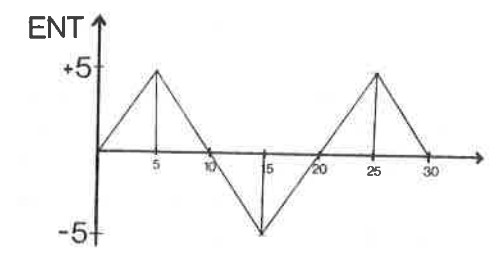
Der Aufbau des Befehls mit Schrittzahl, Schrittweite und Pausezeit ist entsprechend zum ENV, jedoch benötigt man beim ENV meist nur drei Teile, nämlich Ansteigen, Halten und Abfallen, beim ENT hingegen meist fünf.

Eine Art von Vibrato wird durch den folgenden Befehl erzeugt. Vergessen Sie aber nicht, im Programm "oh when the saints" die Soundzeilen so zu ändern, daß der ENT auch angesprochen wird. Der Wert, den Sie einsetzen müssen, ist eins. Er ist die ENTnummer, ähnlich dem Label beim ENV Befehl. Zu beachten ist: Wenn Sie eine negative Nummer hier einsetzen, wird die Hüllkurve wiederholt.

Hier der Befehl, den Sie eingeben müssen:

```
ENT 1,5,1,1,10,-1,1,10,1,1,5,-1,1
```

Wir haben auch zum ENT eine Kurve aufgezeichnet, um Ihnen auch diesen noch grafisch zu verdeutlichen.
Es sind die gleichen Einheiten gewählt, wie bei der Kurve des ENV
Sollten Sie selbst eine Hüllkurve erstellen wollen, können Sie dieses nach dem gleichem Schema wie wir auch tun.



Abschließend zu beiden Hüllkurven noch ein Wort: Es muß gesagt werden, daß die Hüllkurven keine eigentlichen Kurven sind. Man muß Sie eher als Rechteckschwingungen bezeichnen. Diese Tatsache hängt damit zusammen, daß man nicht jeden einzelnen Wert berechnen kann, sondern nur die Eckpunkte der Kurventeile. Der kürzeste Weg zwischen zwei Punkten ist nun mal die Gerade. Mit dieser kann man schlecht eine Kurve zeichnen.

Mittels der beiden Hüllkurvenbefehle können Sie theoretisch jeden Ton so modifizieren, daß Sie ihn kaum noch erkennen können. Das kann für einen Synthysizer von Vorteil sein. Der folgende Soundeditor mag Ihnen dafür als Grundstein dienen. Verändern Sie mit den Cursortasten und dem ENV Befehl die Töne, indem Sie die Werte für die Schrittweite oder Schrittzahl steigen lassen, wenn Sie "Cursor hoch Taste" drücken.

Die leichteste Art, die Hüllkurven wirklich voll zu verstehen, ist es, mit den Werten zu experimentieren, denn rein rechnerisch kann man nicht den wirklichen Klang erfahren.

Wenn Sie ein Instrument simulieren wollen, müssen Sie genau wissen, wie dieses klingt. Ein Saiteninstrument hat meistens einen kleinen Nachhall vom Schwingen der Saiten. Um diesen zu erzeugen, muß beim ENV die dritte Stufe, also das Abfallen, länger sein, als beim Anstieg. Bei Schlaginstrumenten, wie einem Schlagzeug oder eine Pauke, ist dieser Nachhall nicht vorhanden. Hier kann das Abfallen

in einer kürzeren Zeitspanne geschehen als das Ansteigen.

Um aber ein Blasinstrument zu imitieren, ist es schon schwieriger eine entsprechende Hüllkurve zu errechnen. Denn es gibt viele verschiedene Arten, Trompete oder Posaune zu spielen. Das wichtigste aber ist, die Lautstärke abrupt ansteigen zu lassen. Das Abfallen ist abhängig von der Spielart.

3.7 SOUNDEDITOR

Wie Sie sicher schon gemerkt haben, ist das Programmieren von Musik eine recht schwierige Angelegenheit, wenn man nur mit reinen Berechnungen arbeitet. Darum haben wir an dieser Stelle des Kapitels ein Programm geschrieben, das es Ihnen ermöglicht, mittels Tastendruck einen Ton zu erzeugen. Wohlgemerkt einen Ton, nicht einfach ein Geräusch. Die Tastenbelegung entspricht der Tonleiter, also von c bis b, die Halbtöne, das sind die Töne der Tontabelle mit dem "A" vor dem Buchstaben, werden mit der Shifttaste und dem entsprechenden Buchstaben erreicht.

Was können Sie mit diesem Programm alles machen?

1. Melodien programmieren
2. Melodien abspeichern
3. Melodien spielen
4. Melodien laden

1.) Unter diesem Punkt können Sie per Tastendruck, wie oben schon erwähnt, Töne erzeugen. Diese Töne "behält" Ihr CPC und kann Sie auf Abruf wieder spielen. Es besteht die Möglichkeit zwischen neun verschiedenen Oktaven, dies geschieht durch drücken einer Taste zwischen 1 und 9 (die Tastenbelegung wird später auch noch in einer Tabelle erklärt). Durch drücken der Taste "P" können Sie sich die jeweils zuletzt gedrückte Taste auf dem Bildschirm ausgeben lassen. Beim Programmpunkt "Melodien programmieren" besteht jedoch eine Einschränkung hinsichtlich der Tonzahl. Sie ist auf tausend Töne beschränkt, damit der Speicherplatz nicht voll ausgelastet wird und es keine Komplikationen mit dem Abspeichern gibt. Die Anzahl der schon programmierten Töne, wird in der oberen rechten Ecke des Bildschirmes angezeigt.

Die Tondauer können Sie durch längeres drücken der Taste bestimmen. Dabei jedoch entsteht eine kleine Pause beim Programmieren, die aber beim Abspielen der Melodie nicht ins Gewicht fällt.

Um von Punkt eins aus wieder ins Menü zurückzugelangen, müssen Sie die "ENTER"-Taste drücken.

2.) Unter diesem Punkt können Sie eine programmierte Melodie abspeichern. Dazu müssen Sie den Namen Ihrer Melodie eingeben und auf die Anweisung hin den Cassettenrecorder auf Aufnahme stellen. Die Noten werden in Form einer Variablenliste abgespeichert Sie werden mit Punkt "4" wieder geladen. Vermeiden Sie aber, die Melodien zu nahe hintereinander abzuspeichern, da sie sich sonst überschreiben würden. Außerdem ist das Wiederfinden einer Melodie dann einfacher.

3.) Spielen einer gerade programmierten Melodie, bzw. einer neugeladenen Melodie. Hierbei werden die Töne in der Reihenfolge ihres Auftretens abgespielt, jedoch ohne Pausen. Möchten Sie eine Pause mitprogrammieren, müssen Sie beim Punkt "Programmieren" die Leertaste anstatt einer Note drücken.

4.) Laden einer Melodie, die vorher auf Band geschrieben wurde. Die geladene Melodie können Sie dann mit Punkt "3" wieder aufrufen. Es besteht jedoch keine Möglichkeit, die Melodie zu verändern, jedoch kann dies durch kleine Änderungen im Programm erreicht werden. Diese Änderungen dürften Ihnen nach Erklärung des Listings keine Probleme bereiten.

Weitere änderungsbedürftige Stellen sind :
(a) Das Programm arbeitet nur mit einem Tonkanal. Durch zwei weitere Soundbefehle und einer weitere Tastenbelegung können Sie dieses beheben.
(b) Auch gibt es keine Möglichkeit, die Lautstärke vom Programm aus zu ändern, jedoch können Sie ja mit dem Lautstärkeregler am CPC bzw. an Ihrer Stereoanlage diese Veränderung manuell vornehmen.

Im übrigen ist es zum vollständigen Ausnutzen dieses Programmes ratsam, den CPC, an eine Stereoanlage anzuschliessen, da sonst der Klang nicht besonders gut ist.

Es folgt, wie versprochen, eine Tabelle aller Tastenbelegungen und Ihrer Bedeutungen, sowie einigen kurzen Kommentaren.

Taste	Bedeutung	Kommentar
c	Note c	
C	Halbnote c	
d	Note d	
D	Halbnote d	
e	Note e	
f	Note f	
F	Halbnote f	
g	Note g	
G	Halbnote g	
a	Note a	
A	Halbnote a	
b	Note b	
Leertaste	Pause	
p	Tastenausdruck	Beim ersten Mal an, danach aus

1	Oktave -3	Oktavenzahl nach Handbuchtabelle
2	Oktave -2	
3	Oktave -1	
4	Oktave 0	
5	Oktave 1	
6	Oktave 2	
7	Oktave 3	
8	Oktave 4	
9	Oktave 5	Nicht angegeben im Handbuch
ENTER	Ende	Rücksprung zum Menü

```
10 DIM ton(1000)
20 MODE 2
30 LOCATE 5,10:PRINT " Melodie programmieren          :1"
40 LOCATE 5,12:PRINT " Melodie abspeichern            :2"
50 LOCATE 5,14:PRINT " Melodie spielen                :3"
60 LOCATE 5,16:PRINT " Melodie laden                  :4"
70 LOCATE 5,20:INPUT" Ihre Wahl";wa
80 ON wa GOTO 100,490,330,390
90 END
100 CLS
110 PRINT" M E L O D I E   P R O G R A M M I E R E N"
120 n=0
130 FOR toene=1 TO 1000
140 a$=INKEY$:IF a$="" THEN 140
150 a=a+(478 AND a$="c")+(426 AND a$="d")
160 a=a+(379 AND a$="e")+(358 AND a$="f")
170 a=a+(319 AND a$="g")+(284 AND a$="a")
180 a=a+(253 AND a$="b")+(451 AND a$="C")
190 a=a+(402 AND a$="D")+(338 AND a$="F")
200 a=a+(301 AND a$="G")+(268 AND a$="A")
210 IF a$="p" AND pr=0 THEN pr=1:GOTO 140
220 IF a$="p" AND pr=1 THEN pr=0
230 IF a$=CHR$(13) THEN GOTO 20
240 nn=VAL(a$):IF  nn=0  THEN n=n ELSE GOTO 320
250 IF pr=1 THEN PRINT a$;
260 ton(toene)=(a*2^n)
270 SOUND 1,ton(toene)
280 LOCATE 1,1:PRINT toene
290 a=0
300 NEXT
310 GOTO 20
```

```
320 n=4-nn:GOTO 140
330 CLS
340 LOCATE 5,15:PRINT "Melodie wird gespielt"
350 FOR n= 1 TO toene
360 SOUND 1,ton(n)
370 NEXT
380 GOTO 20
390 CLS
400 LOCATE 5,5:PRINT"                    M E L O D I E  L A D
E N"
410 LOCATE 1,20
420 OPENIN"
430 INPUT#9,toene
440 FOR N=1 TO toene
450 INPUT#9,ton(N)
460 NEXT N
470 CLOSEIN
480 GOTO 20
490 CLS
500 LOCATE 5,5:PRINT"                    M E L O D I E  S A V E
 N"
510 INPUT" Wie soll die Melodie heissen?";na$
520 OPENOUT na$
530 PRINT#9,toene
540 FOR n=1 TO toene
550 PRINT#9,ton(n)
560 NEXT
570 CLOSEOUT
580 GOTO 20
```

Erkärungen zum Listing

Zeile 20-90	:	Menü und Auswahl des Modus(1-4)
Zeile 100-310	:	Programmiermodus
Zeile 130	:	Eröffnen der Schleife für die Tonzahl
Zeile 140	:	Abfrage der Tastatur
Zeile 150-230	:	Auswertung der gedrückten Taste
Zeile 240	:	Auswertung, ob Oktave geändert wurde
Zeile 250	:	Ausgabe der gedrückten Taste, wenn gewünscht
Zeile 260	:	Einschreiben des Wertes in die aktuelle Variable des Arrays
Zeile 270	:	Tonerzeugung mittels des Soundbefehls
Zeile 310	:	Rücksprung zum Menü

Zeile 320	: Umstellung der Oktave, Rücksprung zum Programmiermodus
Zeile 330-380	: Spielmodus
Zeile 350	: Eröffnung der Schleife für die Tonzahl
Zeile 360	: Spielen des aktuellen Tones
Zeile 380	: Rücksprung zum MenÜ
Zeile 390-480	: Melodie laden
Zeile 420	: Eröffnen des Eingabefiles von Cassette
Zeile 430	: Lesen der Tonanzahl
Zeile 440	: Eröffnen der Schleife für das Toneinlesen
Zeile 450	: Einlesen der einzelnen Töne
Zeile 470	: Schließen des Eingabefiles von Cassette
Zeile 480	: Rücksprung zum Hauptprogramm
Zeile 490-580	: Abspeichern einer Melodie
Zeile 510	: Eingabe des Namens der Melodie
Zeile 520	: Eröffnen des Ausgabefiles auf Cassette
Zeile 530	: Schreiben der Tonanzahl auf Band
Zeile 540	: Eröffnen der Schleife für das Schreiben der Töne auf Band
Zeile 550	: Schreiben der einzelnen Töne auf Band
Zeile 570	: Schließen des Ausgabefiles auf Cassette
Zeile 580	: Rücksprung zum Hauptprogramm

3.8 BEISPIELPROGRAMME

Mit den folgenden kleinen Demos können Sie Ihren eigenen Programmen den richtigen Sound verleihen. Wir haben solche Beispiele gewählt, die Sie auch ohne großen Aufwand in unsere beiden Spiele mit einbeziehen können. Es dürfte Ihnen, wie bei der Graphik, auch hier nicht schwerfallen, durch kleine Änderungen aus einem der Demos ein neues zu gestalten.

Alle Demos sind nach dem betitelt, was wir beim Anhören der Demos erkannt haben. Es könnte aber durchaus sein, daß Sie etwas ganz anderes heraushören - je nach Vorstellungsgabe und Erfahrungshorizont.

Noch eine kurze Anmerkung: Die Zeilennummern sind nicht bindend, sie können ohne Vorbehalt geändert werden, wenn die Sprungadressen mit geändert werden.

Noch ein kleiner Tip am Rande: Vielleicht können Sie die Demos ja auch mit den Hüllkurven verbessern, oder eine, dem Ton entsprechende Graphik erstellen, z. B. mit selbst definierten Zeichen. Lassen Sie Ihrer Phantasie freien Lauf. Mit dieser Methode erreichen Sie immer noch die besten Ergebnisse.

```
10 REM Polizeisirene
20 FOR n=100 TO 200 STEP 10
30 SOUND 1,n,2
40 NEXT
50 FOR n=200 TO 100 STEP -10
60 SOUND 1,n,2
70 NEXT
80 GOTO 20
```

```
10 REM Besetztzeichen
20 SOUND 1,100,100,15
30 SOUND 1,0,100
40 GOTO 20
```

```
10 REM Weltraumschlacht
20 FOR n=90 TO 125 STEP INT(RND(1)*10)+1
30 SOUND 1,n,2,15
40 NEXT
50 SOUND 1,0,INT(RND(1)*20)
60 GOTO 20
```

```
10  REM Tonleiter
20 FOR n=127 TO 450  STEP 12
30 SOUND 1,n
40 NEXT n
50 FOR n=450 TO 127  STEP -12
60 SOUND 1,n
70 NEXT n
80 GOTO 20
```

```
10 REM Explosion
20 FOR n=15 TO 1 STEP -1
30 SOUND 1,426,40,n,,,1
40 NEXT
```

```
10 REM Alarm
20 FOR n=500 TO 100 STEP -15
30 SOUND 1,n,4
40 NEXT
50 SOUND 1,0,30
60 GOTO 20
```

```
10 REM entfernte Treffer
20 FOR n=100 TO 800 STEP 15
30 SOUND 1,n,2,15
40 NEXT
50 FOR x=1 TO 7 STEP 0.5
60 SOUND 1,400,5,x,,,1
70 NEXT
80 SOUND 1,0,50
90 GOTO 20
```

```
10 REM CPC's Lieblingsmelodie
20 a=INT(RND(1)*3500)+284
30 SOUND 1,a
40 b=INT(RND(1)*3400)+284
50 SOUND 2,b
60 c=INT(RND(1)*3300)+284
70 SOUND 4,c
80 GOTO 20
```

Das nun folgende Programm ist vom Prinzip her gleich aufgebaut, wie das Musikprogramm "Oh when the Saints". Dieses Programm spielt die Melodie "Happy Birthday", ist jedoch auch wieder nur mit Noten und Notenlängen ausgestattet. Zur Verbesserung des Klangs raten wir Ihnen, es mit ENV Und ENT zu verfeinern.

```
10 REM happy birthday
20 READ a,b
30 IF b=-1 THEN END
40 SOUND 1,a,b
50 GOTO 20
60 DATA 319,50,319,50,284,100,319,100,239,100
70 DATA 253,150,319,50,319,50,284,100,319,100
80 DATA 213,100,239,150,319,50,319,50,159,100
90 DATA 190,100,239,100,253,100,284,100,179,50
100 DATA 179,50,190,100,239,100,213,100,239,150
110 DATA 0,-1
```

Erklärung des Listing:

10	Titel
20	Lesen der Noten, bzw. Notenwerte
30	Abfrage, ob Musikstück zu Ende
40	Spielen der Note A mit der Notenlänge B
50	Rücksprung nach 20
60-110	Datas für die Noten und Notenlängen

3.8.1. SOUNDWECKER

Zur Auflockerung der teilweise recht theoretischen Sounderklärungen, haben wir an diese Stelle ein Programm gestellt, das Sie recht gut gebrauchen können, wenn Sie zu den Menschen gehören, die morgens nicht rechtzeitig aufstehen und deshalb öfters den Bus verpassen oder die Eisenbahn. Es handelt sich bei diesem Programm um eine Digitaluhr, die eine Weckvorrichtung hat, von der man sicher geweckt wird. Unsere Digitaluhr hat zusätzlich die Funktion, daß jede Sekunde durch einen leisen Ton und jede Minute durch einen etwas lauteren Ton angezeigt wird. Der größte Vorteil ist, daß jede volle Stunde eine kleine Melodie ertönt, und so der Eindruck bei Ihren Bekannten entsteht, daß Sie eine echte Schweizer Uhr im Hause hätten.

Unsere Uhr ist einfach gehalten, aber mit einigen kleinen Änderungen ist es möglich, aus einem einfachen Wecker einen zweiten Big Ben zu machen. Um diese Änderungen vorzunehmen, benötigen Sie nur die Noten des gewünschten Musikstückes, das der Wecker spielen soll. Auch können Sie sich durch die Melodie wecken lassen, oder einen noch "schrecklicheren" Ton, als den von uns gewählten.
Erklärungen zur Benutzung:

Nach Start des Programmes werden Sie zuerst nach der aktuellen Uhrzeit gefragt, und zwar in der Reihenfolge: Stunden, Minuten, Sekunden.

Diese Werte geben Sie einzeln ein, durch drücken der ENTER-Taste getrennt. Anschließend werden Sie nach der gewünschten Weckzeit gefragt. Die Eingabe erfolgt in der gleichen Reihenfolge, wie bei der Uhrzeit. Sie müssen jedoch beachten, daß, wenn Sie nur ENTER drücken, die Weckzeit auf Null Uhr eingestellt ist, und Sie dann um Null Uhr, insofern das Programm bis dahin noch im Speicher ist, unsanft geweckt werden.

Wollen Sie den Weckton unterbrechen, so müssen Sie eine Taste drücken. Der Ton verstummt und die Uhr läuft normal weiter

Sollte Ihnen auffallen, daß die Abstimmung der Uhr nicht korrekt ist, so müssen Sie in Zeile 150 die Schrittweite des EVERY Befehls ändern. Konkret heißt das, wenn die Uhr nachgeht, verringern Sie den Wert nach EVERY, geht die Uhr vor, erhöhen Sie diesen.

```
5 REM Soundwecker
10 MODE 2
20 INPUT "stunden ";z
30 INPUT "minuten ";y
40 INPUT "sekunden";x
50 INPUT "weckzeit (h)",z2
60 INPUT "weckzeit (m)",y2
70 INPUT "weckzeit (s)",x2
80 MODE 0
90 LOCATE 3,9:PRINT CHR$(150);STRING$(&D,CHR$(154));CHR$(156
)
100 LOCATE 3,10:PRINT CHR$(149):LOCATE 17,10:PRINT CHR$(149)

110 LOCATE 3,11:PRINT CHR$(149):LOCATE 17,11:PRINT CHR$(149)

120 LOCATE 3,12:PRINT CHR$(149):LOCATE 17,12:PRINT CHR$(149)
130 LOCATE 3,13:PRINT CHR$(147);STRING$(&D,CHR$(154));CHR$(1
53)
140 LOCATE 5,5:PRINT"MUSIKWECKER"
150 EVERY 50,2 GOSUB 170
160 GOTO 160
170 SOUND 1,284,5,5:x=x+1
180 IF x=60 THEN x=0:y=y+1:SOUND 2,71,5,6
190 IF y=60 THEN y=0:z=z+1:GOSUB 270
200 IF z=z2 AND x=x2 AND y=y2 THEN GOSUB 320
210 LOCATE 5,11:PRINT z
220 LOCATE 8,11:PRINT":"
230 LOCATE 9,11:PRINT y
240 LOCATE 12,11:PRINT":"
250 LOCATE 13,11:PRINT x
260 RETURN
270 READ no
280 IF no=-1 THEN RETURN
290 SOUND 4,no,30,7
300 GOTO 270
```

```
310 DATA 253,253,169,169,150,150,169,190,190,201,201,225,225
,253, 169,169,190,190,201,201,225,169,169,190,201,201
315 DATA 225,253,253,169,169,150,150,169,190,190,201,201,225
,225,223,-1
320 SOUND 7,100,1000,15
330 IF INKEY$="" THEN GOTO 330
340 SOUND 135,0
350 RETURN
```

Erklärung des Listings:

Zeile 10-80	Setzen des Bildschirmmodes, Eingabe der erforderlichen Werte (Uhrzeit (h,m,s); Weckzeit (h,m,s)
Zeile 90-140	Zeichnen der Umrandung der Uhr
Zeile 150	Durch den EVERY Befehl wird jede Sekunde die Unterroutine ab 170 aufgerufen.
Zeile 170	Erzeugen des Sekundentons und zuzählen einer Sekunde
Zeile 180	Abfrage, ob Minute erreicht, wenn erfüllt, dann erzeugen des Minutentons
Zeile 190	Abfrage, ob Stunde erreicht, wenn erfüllt, dann Sprung zur Unterroutine ab 270
Zeile 200	Abfrage, ob Weckzeit erreicht,wenn erfüllt, dann Sprung zur Unterroutine ab 320
Zeile 210-260	Ausdruck der Uhrzeit in die Umrandung
Zeile 270	lesen der Noten
Zeile 280	Abfrage, ob Ende der Melodie erreicht
Zeile 290	spielen der Melodie, mit allen drei Tonkanälen
Zeile 310	Datazeile mit Notenwerten
Zeile 320	Erzeugen des Wecktons
Zeile 330	an dieser Stelle wartet das Programm, bis eine Taste gedrückt wird.
Zeile 340	Mit dieser Art des Soundbefehles wird der Weckton unterbrochen, es wird ein Halt an die Tonkanäle gesendet

K A P I T E L 4 MASCHINENSPRACHE

4.1 EINFÜHRUNG IN DIE MASCHINENSPRACHE

Diese Einführung ist kein umfassender Maschinensprachekurs. Sie soll Ihnen einen kleinen Einblick in die weiterführende Programmierung Ihres CPC 464 geben. Wie Sie sicher wissen, gibt es einige Probleme bei der Programmierung in Basic, die sich nur durch kleine Maschinenroutinen lösen lassen. Zum Beispiel, das Ausführen einer Sortier- oder Suchroutine, die in Basic manchmal mehrere Minuten oder sogar Stunden benötigt, um eine bestimmte Sache aus einer großen Anzahl von Daten herauszusuchen. Dasselbe Programm in Maschinensprache geschrieben, benötigt nur bis zu einem hundertstel der Zeit, die das Basicprogramm braucht.

Das Gehirn des CPC 464 ist ein Z 80-Baustein, welcher einer der gebräuchlichsten 8-bit Prozessoren in der Homecomputertechnik ist. Dieser Z 80 hat einen Befehlssatz von über 600 Befehlen, für deren vollständige Erklärung in diesem Buch nicht genügend Platz zur Verfügung steht.
Für Ihren CPC gibt es nur eine Möglichkeit Zahlen zu erfassen, nämlich im Binärcode. Binär dargestellte Zahlen bestehen aus den Ziffern "0" und "1", welche in bestimmter Reihenfolge zusammengesetzt jede beliebige Zahl darstellen können. Ein Teil einer solchen Zahl, also eine bestimmte "0" oder "1", heißt BIT. Das Wort Bit kommt aus dem Englischen und steht für Binary Digit (Binärziffer).

Meist sind diese Bits in Achtergruppen angeordnet. Eine solche Achtergruppe heißt dann Byte. Schauen wir ein solches Byte einmal an: 01010101
Ohne Vorwissen kann man sich schwer vorstellen, daß dieses Byte die Zahl "85" darstellt. Um das zu verstehen, muß man wissen, daß das Bit ganz rechts die Wertigkeit 2 hoch 0 repräsentiert, das nächste Bit links davon die Wertigkeit 2

hoch 1, das nächste 2 hoch 2 und das Bit ganz links 2 hoch 7. Wollen wir die Dezimalzahl "85" aus der Binärzahl "01010101" errechnen, so müssen Sie nur das erste Bit ganz rechts, also die "1", mit 2 hoch 0 malnehmen, das zweite Bit, also die "0", mit 2 hoch 1, das dritte Bit mit 2 hoch 2,... und das achte, ganz links liegende Bit, mit 2 hoch 7. Alle diese Ergebnisse zählen Sie nun zusammen und erhalten die Zahl "85". Praktisch sieht das so aus:

```
1 * 2^0 =  1
0 * 2^1 =  0
1 * 2^2 =  4
0 * 2^3 =  0
1 * 2^4 = 16
0 * 2^5 =  0
1 * 2^6 = 64
0 * 2^7 =  0
          ---
Das ergibt zusammengezählt 85
```

In Variablen geschrieben sieht ein Byte wie folgt aus:

```
b7 b6 b5 b4 b3 b2 b1 b0
```

Hier steht b0 für das Bit ganz rechts und b7 für das Bit ganz links.

Die Dezimalzahl errechnet man dann so:

```
b7 * 2^7 + b6 * 2^6 + b5 * 2^5 + b4 * 2^4 +
b3 * 2^3 + b2 * 2^2 + b1 * 2^1 + b0 * 2^0
```

Zum leichteren Rechnen haben wir hier alle Zweierpotenzen vom 2 hoch 0 bis 2 hoch 7 aufgeführt:

2^0=1 2^1=2 2^2=4 2^3=8 2^4=16 2^5=32 2^6=64 2^7=128

Wenn Sie die binäre Darstellung näher betrachten, werden Sie auch verstehen, warum von rechts nach links, von null bis sieben und nicht von eins bis acht nummeriert ist. Dies hängt mit den, zu den Stellen gehörenden, Zweierpotenzen zusammen.
Zur Umrechnung von binären Zahlen in dezimale müssen Sie nur PRINT &X bin eingeben, wobei (bin) für Ihre binäre Zahl steht. Für die Umrechnung von dezimal nach binär müssen Sie die Funktion PRINT BIN$(dez) benutzen. Hier steht (dez) für Ihre dezimale Zahl, die umgerechnet werden soll.

Beispiel:

```
Print &x01010101
85

Print BIN$(85)
01010101
```

Zu beachten ist, daß alle Nullen vor der ersten eins von links meist weggelassen werden.

4.2 DAS HEXADEZIMALE ZAHLENSYSTEM

Ein weiteres sehr wichtiges Zahlensystem ist das hexadezimale oder auch sedezimale System. Es ist wohl die weitverbreiteste Art, den Rechner in Maschinensprache, zu programmieren. Das hexadezimale System basiert auf sechzehn Ziffern: Diese repräsentieren die dezimalen Werte von 0 bis 15.

hex.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
dez.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Der große Vorteil des hexadezimalen Systems ist eine Verkürzung der Eingabelänge, sowie der Eingabezeit von Maschinenprogrammen und Daten. Stellen Sie sich einmal vor, Sie müßten die Zahl "255" eingeben und hätten die Wahl zwischen "11111111" im Binärcode oder "FF" im Hexadezimalcode. Sie würden sicher auch das viel kürzere "FF" bevorzugen, wie fast jeder Programmierer heutzutage.

Wieder zurück zum hexadezimalen System. Der Vorteil dieses Systems liegt darin, daß man ein Byte (also acht Bit) mit zwei Zeichen darstellen kann.

Ist also eine hexadezimale Zahl ein Byte lang, so wird sie folgendermaßen ins Dezimalsystem umgerechnet:

Sie rechnen den ersten Teil der Zahl mal 16 und addieren den zweiten Teil hinzu.
Hier ein Beispiel:

Die Zahl heißt A1. Nehmen Sie den ersten Teil, also das A, mit 16 mal. Das ergibt 160, denn das "A" hat, laut unserer Tabelle, den Wert "10". Addieren Sie 1 hinzu, dann erhalten Sie die Zahl 161.

Bei Zahlen von zwei Byte (also 16 Bit) Länge werden diese in zwei Teile gespalten, in Highbyte und Lowbyte. Nehmen wir die Zahl A1E1 , hier ist A1 das Highbyte und E1 das Lowbyte, verallgemeinert heißt das, das linke Byte einer zwei Bytezahl ist immer das Highbyte und das rechte das Lowbyte. Die Berechnung ist nun recht einfach; zuerst berechnen Sie jedes Byte für sich, wie oben beschrieben, dann nehmen sie das Highbyte mit 256 mal und addieren das Lowbyte.

Auch hierzu folgt ein Beispiel:

> Die Zahl heißt A1E1. Zuerst berechnen wir die Byte
> für sich:
> Highbyte=161;
> Lowbyte=225.
> Nun nehmen Sie das Highbyte mit 256 mal und zählen
> das Lowbyte hinzu.
> Das Ergebnis lautet 41441.

Hier die Rechnung ohne Erklärung:

```
Highbyte(161)*256 +Lowbyte(225)=41441
```

Wenn Sie sich diese Rechnungen ersparen wollen, so hat Ihr CPC 464 eine hervorragende Funktion zur Berechnung von hexadezimalen Zahlen aus dezimalen und umgekehrt. Hier die beiden Funktionen:
Hexadezimal nach Dezimal: PRINT &hex , wobei hex für Ihre hexadezimale Zahl steht.

Dezimal nach Hexadezimal:PRINT HEX$(dez), wobei dez für Ihre dezimale Zahl steht.

Hierbei besteht jedoch eine Einschränkung, es sollten nur hexadezimale Zahlen bis 7FFF mit der ersten Funktion umgerechnet werden, da bei der Umrechnung von größeren Werten 65536 von dem Ergebnis abgezogen wird und das ergibt

eine negative Zahl.

Beispiel:
PRINT &A1E1, erwartet wird das oben schon genannte Ergebnis 41441.
Jedoch wird als Ergebnis -24095 ausgegeben (41441-65536=-24095).

Um dies zu umgehen, können Sie diese kleine Routine benutzen:

```
10 INPUT A$:B$="&"+A$:IF VAL(B$) 0 THEN PRINT VAL (B$)
+65536:END
20 PRINT VAL(B$)
```

Wir haben eine Stringvariable benutzt, damit Sie bei der Eingabe der hexadezimalen Zahl nicht immer ein "&" schreiben müssen.

Prägen Sie sich die Begriffe Highbyte und Lowbyte gut ein, Sie werden sie in späteren Kapiteln öfters finden, denn Speicherstellen, auch Adressen genannt, sind Maschinensprache immer in Highbyte und Lowbyte verschlüsselt. Damit erklärt sich auch, daß die größte Adresse 65535 ist, denn das Highbyte kann als größten Wert nur 255 oder Hexadezimal &FF annehmen, und das Lowbyte ebenso. Somit ergibt sich nach der Formel HIGHBYTE*256 + LOWBYTE als größter Wert 65535.
Zum Abschluß folgt hier noch eine Tabelle aller Zahlen von 0 bis 16 in allen drei System:

Dezimal	Hexadezimal	Binär
1	1	0001
2	2	0010
3	3	0011
4	4	0100

5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

Mit Hilfe dieser Tabelle und dem oben vermittelten Wissen wird es Ihnen sicher nicht mehr schwerfallen, unseren weiteren Erläuterungen zu folgen.

4.3 PROGRAMMIERUNGSTECHNIKEN

Um von Basic aus Maschinensprache zu programmieren, können Sie mehrere Methoden benutzen: Die erste und zeitlich längste ist die, die einzelnen Codes für die Befehle in die Speicherstellen hineinzupoken. Eine weitaus komfortable Art ist die, einen sogenannten Maschinensprachemonitor zu benutzen. Am Ende dieses Kapitels haben wir einen provisorischen Maschinensprachemonitor konzipiert, der Ihnen bei der Programmierung von Maschinenprogrammen eine Hilfe sein wird. Er poked die Werte zwar auch nur in die Speicherstellen, nimmt Ihnen aber die Arbeit des langen Tippens ab. Die ausführliche Erklärung folgt später.

Zur ersten Methode: Wie Sie sicher wissen, schreibt der Befehl POKE X,A den Wert A in die Speicherstelle (Adresse) X. In Maschinensprache sind die Befehle einer bestimmten Adresse zugeordnet, wie die Befehle in Basic den Zeilennummern. Wenn Sie also ein Programm ab Adresse &2000 schreiben wollen, so poken Sie den Wert für den ersten Befehl in die Speicherstelle &2000, die Werte zu dem Befehl, sofern er welche benötigt, in die Speicherstellen &2001 und &2002. Ist der Befehl nur ein Byte lang, so steht in der Speicherstelle &2001 schon der zweite Befehl.

Was hat es mit den Codes auf sich, die zu den Befehlen gehören? Der Rechner versteht, wie Sie wissen nur Binärzahlen, deshalb muß ein Befehl auch aus diesen Binärzahlen bestehen. Jedem Befehl sind eine oder mehrere solcher Zahlen (Codes) zugeordnet. So ergibt sich eine lange Liste von Zahlen für ein Programm, diese Zahlen müssen Sie in der auftretenden Reihenfolge in die Speicherstellen hineinschreiben. Das geschieht durch den Befehl POKE.

Um nun eine Speicherstelle auszulesen, das heißt ihren Wert zu erfahren, müssen Sie den Befehl PEEK (X) anwenden. Er gibt Ihnen den Wert der Speicherstelle X bekannt. Wollen Sie

ein geschriebenes Programm starten, so wird der Befehl CALL X angewendt. Er ruft das Programm ab, der Speicherstelle X auf.

Im folgenden Teil werden auf Sie einige Probleme zukommen, aber keine Angst, wir haben die Maschinensprache auch nicht in drei Tagen gelernt.

Assembler

Der Assemblercode ist eine Programmiersprache, die Abkürzungen für die eigentlichen Maschinensprachebefehle benutzt, diese Abkürzungen heißen MNEMONICS. So steht zum Beispiel für den Befehl "lade",die Abkürzung "ld". Da die Maschine diese Abkürzung aber nicht versteht, benötigt man ein Programm um diese Befehle in Binärzahlen umzuwandeln. Ein solches Programm ist ein Assembler. Mit solch einem Assembler kann man außerdem noch den wichtigen Adressen symbolische Namen geben, sogenannte Labels (englisch:Zeichen). Labels verwenden wir später auch, um den Adressen für die Variablen, die wir benutzen, einen Namen zu geben.

Wie wir unserem Handbuch entnehmen konnten, ist ein Assemblerprogramm von der Firma Amstrad in Vorbereitung. Sollten Sie kein Assemblerprogramm besitzen, müssen Sie jedesmal den zu dem Befehl gehörigen Wert aus der Befehlstabelle im Anhang heraussuchen.

Register und erste Befehle

Zuerst wollen wir einmal den Aufbau der Register der CPU (Gehirn des Rechners) besprechen.

Es gibt mehrere Register, die wichtigsten sind:

--Das a-Register, meist auch Akkumulator genannt. Es wird hauptsächlich zur Speicherung und Verarbeitung aktueller Werte benutzt, das heißt, für Werte die gerade behandelt werden. Das kommt daher, weil das a-Register sehr vielseitig einsetzbar ist.

--Das bc-Register, das bc steht für Bytecounter, weil es oft für Zählaufgaben benutzt wird. Das bc-Register ist ein sogenanntes Doppelregister oder Registerpaar, da man das b-Register, wie auch das c-Register einzeln ansprechen kann. Im Gegensatz zum a-Register, welches nur ein Byte aufnehmen kann, kann das bc-Register zwei Byte aufnehmen und damit Zahlen bis 65535 aufnehmen

--Als drittes gibt es das hl-Registerpaar, auch High-Low Registerpaar genannt. Hier gilt das gleiche wie für das bc-Registerpaar. Jedes Register kann einzeln angesprochen werden, doch nutzt man es besser immer als 16 Bit (2 Byte) Register, denn die CPU bietet dieses Register geradezu an zur Speicherung von Adressen an, die bekanntlich mit 2 Byte dargestellt werden. (Highbyte und Lowbyte).

--Als letztes wichtiges Registerpaar möchten wir das de-Registerpaar ansprechen. Jeder Könner der Maschinen- sprache wird fragen, was ist mit den ix und iy Registern oder mit dem Stapelzeiger oder dem Programmcounter. Für die Könner unter Ihnen ist dieses Kapitel auch nicht gedacht, sondern für die vielen Einsteiger im Maschinensprachebereich. So, nun aber zum de-Register, es ist wie das bc oder das hl Registerpaar auch einzeln einzusetzen. Es ist ein Hilfsregister um 16 Bit Zahlen zu speichern, die in den anderen Registerpaaren nicht sinnvoll wären oder wenn diese belegt sind. Oder auch ganz einfach deshalb, weil mancher Befehl drei Registerpaare benötigt.

Wenn man sich den Aufbau der CPU ansieht, so fragt man sich, wie man einen bestimmten Wert aus dem Speicher in die Register bekommt oder von einem Register in ein anderes. Dies geschieht durch den einfachen Befehl "ld" (lade), er ist der wichtigste Befehl im Z 80 Befehlssatz. Ein Beispiel dazu:

Wenn Sie den Akkumulator (a-Register) mit dem Wert &FF (dezimal 255) laden wollen, so schreiben Sie den Wert für: ld a,n , wobei Sie für n den Wert &FF einsetzen müssen. Das n steht für eine direkte Eingabe, die nur ein Byte lang ist. Steht ein "nn", so bedeutet das ebenfalls eine direkte Eingabe, aber zwei Byte lang und in High- und Lowbyte aufgeschlüsselt. Steht eine Klammer um die beiden n, so bedeutet dies, daß das Register mit dem Inhalt der Speicherstelle nn geladen wird.

Beispiel: ld hl,(&A000) bedeutet, daß das hl Registerpaar mit dem Inhalt der Speicherstelle &A000 geladen wird.

Wollen Sie zum Beispiel das c Register mit dem Inhalt des Akkumulators laden, so schreiben Sie ld c,a. Hier geht der Inhalt des c-Registers verloren, der des Akkumulators jedoch nicht.

Um Maschinensprache zu programmieren, benötigt man einen sicheren Speicherbereich, der nicht von einem Basicprogramm überschrieben werden kann. Die folgende Befehlsfolge schafft über dem Basicspeicher einen freien Bereich, indem Sie die noch folgenden Routinen ablegen können:
Geben Sie direkt ein :

MEMORY &1FFF:PRINT HIMEM:NEW

Sollten Sie selbstdefinierte Zeichen programiert haben, so löschen Sie diese durch kurzes Ausschalten, bevor Sie diese

Befehlsfolge nutzen, denn es könnten sonst Komplikationen auftreten.

Damit haben Sie einen freien Bereich von mehr als 34 Kilobyte Länge für ihre Maschinenprogramme, jedoch nur noch 7.5 Kilobyte für Basic.
Geben Sie nun ein POKE &2000,&3E:POKE &2001,&FF:POKE &2002,&C9
Das POKE bewirkt das Einschreiben der Befehle in die Speicherstellen
Die drei Werte nach dem Poke sind die Adressen in die die Befehle geschrieben werden. Sie sind den Zeilennummern in Basic ähnlich. Der Wert &3E steht für den oben schon erklärten Befehl ld a,n. Der Wert &FF ist der Wert der für n eingesetzt wird. Der Wert &C9 steht für RET, was RETURN bedeutet, also ein Rücksprung von der Routine nach Basic.

Starten Sie nun das Programm mit CALL (&2000)
Wie Sie sicher wissen, ruft man mit CALL eine Maschinenspracheroutine auf.

Wenn Sie die Routine gestartet haben, meldet sich der Rechner nach einem Augenblick mit "READY", und es ist nicht ersichtlich, was passiert ist, um genau zu sagen, es ist keine Veränderung zu sehen. Jedoch haben Sie durch dieses Programm den Akkumulator mit &FF geladen, was auf den Computer keine direkte Wirkung zeigt.

Versuchen Sie nun folgendes :ld bc,&A1F1. Geben Sie also ein POKE &2000,&1 (ld bc,nn), POKE &2001,&F1 (Lowbyte der Zahl), POKE &2002,&A1 (Highbyte der Zahl), POKE &2003,&C9 (RET).

Starten Sie wieder mit CALL &2000, wieder ist keine Veränderung zu sehen, jedoch haben Sie nun das bc Registerpaar mit &A1F1 geladen. Beiläufig haben Sie schon eine grundlegende Regel der Programmierung benutzt, und

zwar, daß nach dem Befehl, bei einem 16 Bit Operanden, immer zuerst das Lowbyte eingetragen wird und dann das Highbyte.

Lassen Sie uns zu etwas sinnvollerem kommen, schreiben wir ein Additionsprogramm für Einbytezahlen.

Sagen wir, der erste Summand soll ADD1 heißen, der zweite ADD2 und das Resultat RES. Lassen wir ADD1 in der Speicherstelle &3000 stehen, ADD2 in &3002 und das Resultat in &3004. Wir müssen nun eine Möglichkeit finden, die beiden Werte zu addieren. Zuerst bringen wir den ersten Summanden (ADD1) in den Akkumulator und dann die Adresse des zweiten (ADD2) in das hl-Register. Das erfolgt durch die Befehle ld a,(&3000), ld hl,&3002. Wir laden das hl-Register nicht sofort mit ADD2, um Ihnen die Möglichkeiten der Programmierung zu zeigen.

Wir benötigen noch einen Befehl zum Addieren der beiden Summanden. Dieser ist add a,(hl). Dieser Befehl bewirkt, daß der Inhalt der Speicherstelle, die durch das hl-Register adressiert ist, also der Inhalt der Speicherstelle in der ADD2 steht, zum Inhalt des Akkumulators hinzugezählt wird. Schließlich müssen wir noch das Ergebnis in die Speicherstelle &3004 abspeichern. Das geschieht durch den Befehl ld (&3004),a. Am Schluß steht noch RET, um wieder zum Basic zurückzugelangen. Folgend haben wir das Programm noch einmal untereinander ohne Kommentar ausgedruckt, hinter den Assemblerbefehlen stehen die hexadezimalen Werte, die Sie einpoken müssen.

```
ld a,(&3000)        &3A &00 &30
ld hl,&3002         &21 &02 &30
add a,(hl)          &86
ld (&3004),a        &32 &04 &30
RET                 &C9
```

Um von Basic aus dieses Programm einzugeben, benutzen Sie bitte
die folgende Basicroutine:

```
10 FOR N=&2000 TO &200A
20 INPUT WERT
30 POKE N,WERT
40 NEXT
```

Nach dem Start dieser Routine geben Sie die oben angegebenen Werte ein. Sie müssen nur noch zwei Zahlen bis (bis maximal 127) in die Speicherstellen &3000 und &3002 schreiben, und zwar mit POKE &3000,ADD1:POKE &3002,ADD2.

Starten Sie das Maschinenprogramm mit CALL &2000.
Nach dem "READY" können Sie mit PEEK (&3004) das Ergebnis (RES) erfahren.

Die Zahlen dürfen deshalb nur bis 127 groß sein, damit das Ergebnis nicht größer als 255 wird, da es sonst nicht mehr in ein Byte paßt.

Jetzt wollen wir uns den Ablauf der Addition im Binärcode an einem Beispiel ansehen:
ADD1 ist 127 und ADD2 ist 64

```
ADD1=  01111111
ADD2=  01000000

RES =  10111111 oder dezimal 191
```

Sie sehen sicher sofort, daß die Addition von Binärzahlen nicht schwer ist, wenn ein Bit gesetzt ist und das entsprechende Bit im zweiten Summanden nicht, so wird im Ergebnis dieses Bit gesetzt, ebenso umgekehrt. Sind beide entsprechenden Bits gesetzt, so wird das Bit im Summanden nicht gesetzt, aber es entsteht ein Übertrag (CARRY), dieser wird im Bit links daneben verwertet (siehe Bit 6 und 7). Erfolgt ein solcher Übertrag im ganz linken Bit, also Bit 7,

so wird er bei einer acht-bit Operation nicht verwertet. Jedoch bei einer sechzehn-bit Addition. Diese folgt als nächste. Daraus erklärt sich auch, daß man nur Zahlen bis 127 benutzen darf, da sonst ein solcher Übertrag auftreten würde.

Eine sechzehn-bit Addition ist im Prinzip das gleiche wie eine acht-bit Addition. Jedoch müssen wir hierzu mehr Speicherstellen benennen und zwei neue Befehle einführen.

Zuerst wollen wir die Speicherstellen für die Summanden benennen. Für den ersten Summanden (ADD1) stehen die Speicherstellen &3001, später ADR1 genannt, und &3000, später ADR1-1 genannt, zur Verfügung, für den zweiten Summanden (ADD2), die Speicherstellen &3003 (ADR2) und &3002 (ADR2-1). Das Ergebnis wird in &3005 (RES) und &3004 (RES-1) abgelegt.

Nun zum Programm:
Zuerst werden die beiden Lowbytes der Zahlen addiert und das Ergebnis in RES abgespeichert. Tritt ein Übertrag auf, so wird das Carryflag (Übertragszeichen) gesetzt. Dieses wird im zweiten Teil des Programmes verwertet.

In diesem zweiten Teil werden die Highbytes zusammengezählt und ein eventueller Übertrag wird verwertet. Hier das Programm in symbolischer Schreibweise:

```
ld a,(ADR1)      Lade das Lowbyte von ADD1 in
                 den Akkumulator
ld hl,ADR2       Lade die Adresse des Lowbytes
                 von ADD2 nach hl
add a,(hl)       Addiere die Lowbytes
ld (RES),a       Speichere das Lowbyte des
                 Ergebnisses nach RES
ld a,(ADR1-1)    Lade das Highbyte von ADD1
                 in den Akkumulator
```

```
dec hl            Ziehe eins von hl ab
adc a,(hl)        Addiere die Highbytes und
                  den Übertrag
ld (RES-1),a      Speichere das Highbyte des
                  Ergebnisses nach RES-1
```

Der erste neue Befehl ist dec hl, er bewirkt das gleiche wie ld hl,ADR-1, denn dadurch, daß man von hl eins abzieht, steht in hl nun ADR1-1 anstatt ADR1. Der Vorteil bei dec hl ist der, daß der Befehl nur ein Byte lang ist und deshalb viel eleganter und schneller ist.

Der zweite neue Befehl ist adc a,(hl), er bewirkt im Prinzip das gleiche wie add a,(hl), jedoch wird bei adc a,(hl) auch der Übertrag aus dem ersten Teil berücksichtigt.

Nun aber zum praktischen Teil: Nachfolgend steht das Programm mit eingesetzten Werten und dahinter wieder die hexadezimalen Werte, die Sie eingeben müssen.

```
ld a,(&3001)      &3A &1 &30
ld hl,&3003       &21 &3 &30
add a,(hl)        &86
ld (&3005),a      &32 &5 &30
ld a,(&3000)      &3A 00 &30
dec hl            &2B
adc a,(hl)        &8E
ld (&3004),a      &32 &4 &30
RET               &C9
```

Um dieses Programm einzugeben, benötigen Sie wieder die Schleife, jedoch müssen Sie die Zeile 10 umändern in 10 FOR N=&2000 TO &2012.

Tragen Sie nun Ihren ersten Summanden in die Speicherstellen &3000 und &3001 ein, und zwar das Highbyte in &3000 und das Lowbyte in &3001, ebenso den zweiten Summanden in &3002 und

&3003. Nehmen wir an, der erste Summand habe den Wert &5F4C und der zweite &8B5A, so schreiben Sie folgendes:

```
POKE &3000,&5F:POKE &3001,&4C:POKE &3002,&8B:POKE &3003,&5A
```

Starten Sie nun das Maschinenprogramm mit CALL(&2000).

Nach dem "READY" geben Sie ein:

```
PRINT PEEK(&3004)*256+PEEK(&3005).
```

Nun steht das Ergebnis der Addition auf dem Bildschirm, bei unseren Zahlen lautet es 60070. Dieses Ergebnis kann auch nachgeprüft werden, indem Sie 24396 (&5F4C) und 35674 (&8B5A) addieren.

Bei einer Additon von zwei 16-Bit Zahlen, deren Lowbytes zusammen größer als &FF sind, entsteht also ein Übertrag. Dieser Übertrag wird im Carryflag abgespeichert. Dieses Carryflag ist ein Bit des f-Registers, welches wir am Anfang noch nicht besprochen hatten. Dieses f-Register beinhaltet alle wichtigen Informationen , die die CPU braucht, so zum Beispiel, ob ein Übertrag stattgefunden hat oder ob das b-Register gleich null ist. Wenn also ein Übertrag stattfindet, wird im f-Register das Carryflag gesetzt. Dieser Übertrag wird dann in Bit 0 des Highbytes des Ergebnisses verrechnet. Um dieses zu verdeutlichen, folgt an dieser Stelle ein Beispiel dazu:

```
     Highbyte       Lowbyte

ADD1=00010000 (&10) 10000000 (&80)

ADD2=00001000 (&08) 10000001 (&81)

RES =00011000 (&18) 00000001 (&01)
```

Wie Sie sehen, sind bei einer Rechnung ohne Übertrag 256 verloren gegangen. Wenn man nun das gesetzte Carryflag berücksichtigen will, muß man zu dem Highbyte eine eins hinzuzählen. Dann wird aus dem falschen Ergebnis &1801 das richtige Ergebnis &1901.

Es ist also unerläßlich, bei einer 16 bit Rechnung das Carryflag mit in die Rechnung einzubeziehen. Das haben wir in unserem Programm mit dem Befehl adc a,(hl) getan, damit das Ergebnis, insofern ein Übertrag vorliegt, nicht um 256 zu klein ist.

Zurück zu etwas Einfacherem. Wir wollen Punkte auf dem Bildschirm setzen. Dies ist möglich mit dem Befehl PLOT X,Y. Man kann es aber auch erreichen, indem man in den Bildschirmspeicher hineinpoked. Dieser beginnt bei &C000 und endet bei &FFFF. Schreiben Sie folgendes Programm :

```
10 MODE 2
20 FOR N= &C000 TO &FFFF
30 POKE N,&FF
40 NEXT
```

Sie sehen nach dem Start des Programms, wie sich der Bildschirm füllt, jedoch nicht Zeile für Zeile, sondern es wird erst eine Zeile gefüllt, dann werden 7 Zeilen ausgelassen und die achte wieder gezeichnet. Eine Zeile ist hier so hoch wie ein Plotpunkt. Dies alles hängt mit dem Aufbau der Zeichen zusammen.

Um das folgende Maschinenprogramm vollständig zu verstehen, müssen Sie sich klarmachen, daß man eine solche Schleife auch in der folgenden Form erstellen kann:

```
10 N=N+1
20 IF N=&FFFF THEN END
```

Es muß auch eine Möglichkeit geben, die obige Schleife in

Maschinensprache zu schreiben. Diese haben wir nachfolgend zum Teil verwirklicht, es werden nämlich nur die ersten drei Zeilen gefüllt. Der Grund dafür ist, daß wir Sie nicht mit zu vielen neuen Befehlen überfordern.
Wir müssen also eine Speicherstelle mit &FF laden. Das wird wieder durch einen ld Befehl bewirkt. Danach müssen wir jedoch --e nächste Speicherstelle laden. Wenn wir nicht alle Speicherstellen mit einem eigenen Ladebefehl laden wollen, müssen wir eine Schleife schreiben. Dafür bietet sich das b-Register geradezu an, da man es leicht überwachen kann.

Die folgenden Variaben werden benutzt:

Laenge	Länge des zu füllenden Bereiches
ADR1	Anfangsadresse zum Füllen

Zuerst müssen wir die Schleife definieren. Dies geschieht durch den Befehl ld b, Länge.. Dann laden wir den Akkumulator mit dem Wert &FF. Danach laden wir hl mit ADR1, in der Maschinensprache heißt das, wir setzen einen Zeiger auf die Startadresse. Anschließend laden wir die Adresse, die durch das hl-Register bezeichnet ist, mit dem Inhalt des Akkumulators, also &FF. Jetzt müssen wir hl um eins erhöhen, damit das Register auf die nächste Adresse zeigt. Wir ziehen vom b-Register eine eins ab, um dem Rechner zu signalisieren, daß nur noch so viele Punkte gesetzt werden müssen, wie das b-Register anzeigt. Das heraufzählen wird durch den Befehl inc hl, das herabzählen durch den Befehl dec b. Der nächste Befehl muß nun ein bedingter Sprung sein. Ein bedingter Sprung entspricht einem "IF...THEN GOTO..." im Basic. Wenn also eine Bedingung erfüllt ist, erfolgt ein Sprung. Dieser Sprung muß zum Befehl ld (hl),a erfolgen, da sonst immer nur die erste Speicherstelle geladen wird. Die Bedingung für unseren Sprung muß "NICHT NULL" heißen, denn es soll ja solange gesprungen werden, bis das b-Register ungleich null ist. Der Befehl heißt für unsere Bedingung jp nz, Adresse, das nz steht für not zero (nicht null). Als

letztes müssen wir noch ein RET schreiben, um wieder zum Basic zurückzukehren. Das ganze Programm sieht wie folgt aus:

```
ld b,&C0        &6 &C0
ld a,0          &3E &FF
ld hl,&C000     &21 &0 &C0
ld (hl),a       &77
inc hl          &23
dec b           &5
jp nz,&2007     &C2 &7 &10
RET             &C9
```

Geben Sie die Werte in einer Schleife, wie vorher schon beschrieben, ein. Die Schleife muß nun von &2000 bis &200D gehen.

Bevor Sie das Programm starten, schalten Sie den Computer mit MODE 2 noch einmal in den 80 Zeichenmodus um. Auch wenn Sie sich im 80 Zeichenmodus befinden, schalten Sie Ihn trotzdem noch einmal um, damit keine Komplikationen auftreten.

Starten Sie bitte nun mit CALL &2000, es werden nun drei Zeilen gezeichnet.

Der Jump- (Sprung-) Befehl springt zu der Adresse, an der das ld (hl),a steht. Wie dieses technisch vor sich geht, ist nicht so einfach. Wir müssen hierfür das PC-Register (Programmcounter) einführen. Dieses PC-Register beinhaltet die Adresse des Befehles, der ausgeführt werden soll. Schreibt man in dieses PC-Register nun eine neue Adresse ein, so arbeitet der Rechner den Befehl an der eingegebenen Adresse ab. Eine weitere Möglichkeit, einen Sprung zu berechnen, ist es, vom Programmcounter den Wert ab- bzw. aufzurechnen, der benötigt wird, um die entsprechende Adresse zu erreichen. Diese Art des Sprunges heißt relativ,

da nicht die wirkliche, absolute Adresse in den PC-Counter geladen wird, sondern diese durch Abziehen oder Zuzählen eines Wertes erreicht wird. Der relative Sprung wird durch den Befehl jr e ausgedrückt, wobei das e für den Faktor steht der zum Programmcounter hinzugezählt bzw. abgezogen wird. Jedoch ist die Berechnung eines solchen Sprunges noch etwas schwieriger als hier erklärt. Wie schon gesagt, dieses Kapitel soll nur eine Einführung darstellen, und es sollen nur einige Grundbegriffe geklärt werden.

Zeichen/Befehlssatz

Nachfolgend finden Sie eine Tabelle mit den Zeichen und Mnemonics sowie deren HEx und Dez-Code Ihres CPC

Kode	Charak.	Hex	Z80 Assembler	-nach CBh	-nach EDh
0	Null	00	nop	rlc b	
1	SOH	01	ld bc,NN	rlc c	
2	STX	02	ld (bc),a	rlc d	
3	ETX	03	inc bc	rlc e	
4	EOT	04	inc b	rlc h	
5	ENQ	05	dec b	rlc l	
6	ACK	06	ld b,N	rlc (hl)	
7	BEL	07	rlca	rlc a	
8	BS	08	ex af,af"	rrc b	
9	HT	09	add hl,bc	rrc c	
10	LF	0A	ld a,(bc)	rrc d	
11	VT	0B	dec bc	rrc e	
12	FF	0C	inc c	rrc h	
13	CR	0D	dec c	rrc l	
14	SO	0E	ld c,N	rrc (hl)	
15	SI	0F	rrca	rrc a	
16	DLE	10	djnz DIS	rl b	
17	DC 1	11	ld de,NN	rl c	
18	DC 2	12	ld (de),a	rl d	
19	DC 3	13	inc de	rl e	
20	DC 4	14	inc d	rl h	
21	NAK	15	dec d	rl l	
22	SYN	16	ld d,N	rl (hl)	
23	ETB	17	rla	rl a	
24	CAN	18	jr DIS	rr b	
25	EM	19	add hl,de	rr c	
26	SUB	1A	ld a,(de)	rr d	
27	ESC	1B	dec de	rr e	
28	FS	1C	inc e	rr h	

29	GS	1D	dec e	rr l	
30	RS	1E	ld e,N	rr (hl)	
31	US	1F	rra	rr a	
32	SP	20	jr nz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	CHR$(35)	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld,h,N	sla (hl)	
39	Chr$(39)	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	Chr$(45)	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	Chr$(60)	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	Chr$(62)	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	Chr$(64)	40	ld b,b	bit 0,b	in b,(c)
65	A	41	ld b,c	bit 0,c	out (c),b

66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld(NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	retn
70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	
72	H	48	ld c,b	bit 1,b	inc c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c.l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	
80	P	50	ld d,b	bit 2,b	in d,(d)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 2,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	Chr$(91)	5B	ld e,e	bit 3,e	ld de,(NN)
92	Chr$(92)	5C	ld e,h	bit 3,h	
93	Chr$(93)	5D	ld e,l	bit 3,l	
94	^	5E	ld e,(hl)	bit 3,(hl)	im 2
95	Chr$(95)	5F	ld e,a	bit 3,a	
96	Chr$(96)	60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	

103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld de,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld (hl),b	bit 6,b	
113	q	71	ld (hl),c	bit 6,c	sbc hl,sp
114	r	72	ld (hl),d	bit 6,d	ld (NN),sp
115	s	73	ld (hl),e	bit 6,e	
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123		7B	ld a,e	bit 7,e	ld sp,(NN)
124	G	7C	ld a,h	bit 7,h	
125	R	7D	ld a,l	bit 7,l	
126	A	7E	ld a,(hl)	bit 7,(hl)	
127	P	7F	ld a,a	bit 7,a	
128	H	80	add a,b	res 0,b	
129	I	81	add a,c	res 0,c	
130	C	82	add a,d	res 0,d	
131		83	add a,e	res 0,e	
132	–	84	add a,h	res 0,h	
133		85	add a,l	res 0,l	
134	Z	86	add a,(hl)	res 0,(hl)	
135	E	87	add a,a	res 0,a	
136	I	88	adc a,b	res 1,b	
137	C	89	adc a,c	res 1,c	
138	H	8A	adc a,d	res 1,d	
139	E	8B	adc a,e	res 1,e	

140	N	8C	adc a,h	res 1,h	
141		8D	adc a,l	res 1,l	
142	+	8E	adc a,(hl)	res 1,(hl)	
143		8F	adc a,a	res 1,a	
144	T	90	sub b	res 2,b	
145	O	91	sub c	res 2,c	
146	K	92	sub d	res 2,d	
147	E	93	sub e	res 2,e	
148	N	94	sub h	res 2,h	
149	S	95	sub l	res 2,l	
150		96	sub (hl)	res 2,(hl)	
151		97	sub a	res 2,a	
152		98	sbc a,b	res 3;b	
153		99	sbc a,c	res 3,c	
154		9A	sbc a,d	res 3,d	
155		9B	sbc a,e	res 3,e	
156		9C	sbc a,h	res 3,h	
157		9D	sbc a,l	res 3,l	
158		9E	sbc a,(hl)	res 3,(hl)	
159		9F	sbc a,a	res 3,a	
160		A0	and b	res 4,b	
161		A1	and c	res 4,c	cpi
162		A2	and d	res 4,d	ini
163		A3	and e	res 4,e	outi
164		A4	and h	res 4,h	
165		A5	and l	res 4,l	
166		A6	and (hl)	res 4,(hl)	
167		A7	and a	res 4,a	
168		A8	xor b	res 5,b	ldd
169		A9	xor c	res 5,c	cpd
170		AA	xor d	res 5,d	ind
171		AB	xor e	res 5,e	outd
172		AC	xor h	res 5,h	
173		AD	xor l	res 5,l	
174		AE	xor (hl)	res 5,(hl)	
175		AF	xor a	res 5,a	
176		B0	or b	res 6,b	ldir

177	B1	or c	res 6,c	cpir
178	B2	or d	res 6,d	inir
179	B3	or e	res 6,e	otir
180	B4	or h	res 6,h	
181	B5	or l	res 6,l	
182	B6	or (hl)	res 6,(hl)	
183	B7	or a	res 6,a	
184	B8	cp b	res 7,b	lddr
185	B9	cp c	res 7,c	cpdr
186	BA	cp d	res 7,d	indr
187	BB	cp e	res 7,e	otdr
188	BC	cp h	res 7,h	
189	BD	cp l	res 7,l	
190	BE	cp (hl)	res 7,(hl)	
191	BF	cp a	res 7,a	
192	C0	ret nz	set 0,b	
193	C1	pop bc	set 0,c	
194	C2	jp nz,NN	set 0,d	
195	C3	jp NN	set 0,e	
196	C4	call nz,NN	set 0,h	
197	C5	push bc	set 0,l	
198	C6	add a,N	set 0,(hl)	
199	C7	rst 0	set 0,a	
200	C8	ret z	set 1,b	
201	C9	ret	set 1,c	
202	CA	jp z,NN	set 1,d	
203	CB		set 1,e	
204	CC	call z,NN	set 1,h	
205	CD	call NN	set 1,l	
206	CE	adc a,N	set 1,(hl)	
207	CF	rst 8	set 1,a	
208	D0	ret nc	set 2,b	
209	D1	pop de	set 2,c	
210	D2	jp nc,NN	set 2,d	
211	D3	out N,a	set 2,e	
212	D4	call nc,NN	set 2,h	
213	D5	push de	set 2,l	

214	D6	sub N	set 2,(hl)
215	D7	rst 16	set 2,a
216	D8	ret c	set 3,b
217	D9	exx	set 3,c
218	DA	jp c,NN	set 3,d
219	DB	in a,N	set 3,e
220	DC	call c,NN	set 3,h
221	DD		set 3,l
222	DE	sbc a,N	set 3,(hl)
223	DF	rst 24	set 3,a
224	E0	ret po	set 4,b
225	E1	pop hl	set 4,c
226	E2	jp po,NN	set 4,d
227	E3	ex (sp),hl	set 4,e
228	E4	call po,NN	set 4,h
229	E5	push hl	set 4,l
230	E6	and N	set 4,(hl)
231	E7	rst 32	set 4,a
232	E8	ret pe	set 5,b
233	E9	jp (hl)	set 5,c
234	EA	jp pe,NN	set 5,d
235	EB	ex de,hl	set 5,e
236	EC	call pe,NN	set 5,h
237	ED		set 5,l
238	EE	xor N	set 5,(hl)
239	EF	rst 40	set 5,a
240	F0	ret p	set 6,b
241	F1	pop af	set 6,c
242	F2	jp p,NN	set 6,d
243	F3	di	set 6,e
244	F4	call p,NN	set 6,h
245	F5	push af	set 6,l
246	F6	or N	set 6,(hl)
247	F7	rst 48	set 6,a
248	F8	ret m	set 7,b
249	F9	ld sp,hl	set 7,c
250	FA	jp m,NN	set 7,d

251	FB	ei	set 7,e
252	FC	call m,NN	set 7,h
253	FD		set 7,l
254	FE	cp N	set 7,(hl)
255	FF	rst 56	set 7,a

Maschinenprogramm zum Auslesen des ROMs

```
AB00                        ORG  &AB00
AB00 DF                     RST  3
AB01 04 AB                  DW   TABLE
AB03 C9                     RET
               ;
AB04 07 AB     TABLE        DW   READROM
AB06 FC                     DB   &FC ; Roms selektieren
               ;
AB07 3A 00 00  READROM      LD   A,0 ; Byte lesen
AB0A 32 0E AB               LD   BYTE,A ; und abspeichern
AB0D C9                     RET
               ;
AB0E           BYTE         DS   1    ; Platz für gelesenes Byte
```

Zum Auslesen des ROMs des CPC 464 wird eine Routine des Betriebssystems benutzt. Über den Restart-Vektor 3, der einem Unterprogrammaufruf an Adresse &18 entspricht, läßt sich eine Routine sowohl im RAM als auch im ROM aufrufen. Dazu muß direkt hinter dem RST 3 Befehl die Adresse eines 3-Byte-Vektors stehen für einen sogenannten FAR CALL. Dieser Vektor enthält zuerst die Adresse der Routine, die ausgeführt werden soll und im nächsten Byte die Information darüber, ob ROM oder RAM ausgewählt werden soll. Die Tabelle enthält die entsprechenden Werte.

Wert	Selektierter Speicher
&00 - &FB	Expansion-ROM
&FC	Betriebssystem + BASIC
&FD	BASIC
&FE	Betriebssystem
&FF	RAM

Mit einem Wert von &FC werden sowohl das untere (Betriebssystem) als auch das obere (BASIC) ROM ausgewählt. Für die Routine selbst, die im RAM steht, hat dies keine Bedeutung, da ja im Adreßbereich von &4000 bis &BFFF nur RAM existiert. Die Auswahl des zu lesenden Bytes geschieht einfach dadurch, daß Lo- und Hi-Byte hinter den LD-Befehl an der Adresse READROM vor dem Aufruf der Routine gepokt werden.

Programmbeschreibung zum Mini-Monitor

100 Herabsetzen der Speicherobergrenze für BASIC, Farbzuordnungen und 40-Zeichen-Modus auswählen

110 Einlesen der Befehle in die Variable cmd$

120 Maschinenprogramm laden

130-140 Ausgabe der Überschrift. Für das Unterstreichen wird der Transparent-Modus benutzt.

150-220 Der Befehlssatz und die entsprechenden Buchstaben zum Aufrufen der Befehle werden angezeigt.

230-260 Festlegung des Ausgabemediums für den M-Befehl. Durch Eingabe von ´B´ oder ´D´ können Bildschirm oder Drucker ausgewählt werden. Die entsprechenden Buchstaben werden blinkend dargestellt (PEN 3).

270-300 Abhängig von der Eingabe wird bei Druckausgabe die Anzahl der Bytes pro Zeile (l) auf 16 und die Kanalnummer (c) auf 8 gesetzt. Wurde Bildschirmausgabe gewählt, können Sie noch zwischen 40- und 80-Zeichen-Modus (1 oder 2) auswählen.

310-330 Dies ist die Hauptschleife, in der auf die Eingabe eines Befehls gewartet wird. Das eingegebene Zeichen wird mit Kommandos verglichen und bei Übereinstimmung wird das zugehörige Unterprogramm aufgerufen.

350-390 Hex-Dump. Nach Ausgabe der Adresse werden 1 (8 oder 16) Bytes als zweistellige Hexzahlen ausgegeben. Soll vom RAM gelesen werden, so kann dies einfach durch PEEK geschehen. Soll der Inhalt des ROMs dargestellt werden, so wird mittels GOSUB 890 das ROM gelesen.

400-470 ASCII-Dump. Beim Ausgeben der zugehörigen ASCII-Zeichen wird zunächst unterschieden, ob auf Drucker oder Bildschirm ausgegeben werden soll. Bei Druckausgabe wird das oberste Bit des Zeichens gelöscht. Handelt es sich um ein Kontrollzeichen oder um den Kode für DEL, so wird ersatzweise der ASCII-Kode für den Punkt genommen. Bei der Ausgabe auf den Bildschirm wird im 40-Zeichen-Modus auf die zweite Farbe umgeschaltet und sämtliche Zeichen einschließlich der Kontrollzeichen (durch CHR$(1)) werden als darstellbare Zeichen ausgegeben.

490-520 Diese Routine dient zur Änderung von Speicherinhalten. Nachdem Sie eine Adresse vorgegeben haben, können Sie den neuen Inhalt eingeben. Danach wird automatisch die folgende Adresse vorgegeben und auf die nächste Eingabe gewartet. Diesen Modus können Sie beenden, indem Sie kein gültiges Hex-Zeichen, z.B. einen Punkt eingeben.

540 Diese beiden Unterprogrammaufrufe behandeln den M-Befehl. Mit GOSUB 650 werden Start- und Endadresse des anzuzeigenden Bereichs eingegeben, während der

zweite Aufruf die Anzeige übernimmt.

560-570 Diese Routine wandelt eine eingegebene Hexziffernfolge in einen numerischen Wert um und erspart Ihnen das Voranstellen von ´&´.

590-630 Zum Abspeichern eines RAM-Bereichs auf Kassette dient diese Routine. Dazu müssen Sie die Grenzen dieses Bereichs sowie den Namen, unter dem dies geschehen soll, angeben.

650-680 Zur Eingabe eines Speicherbereichs dient diese Routine. Die Grenzen werden in den Variablen a und b übergeben.

700-730 Wollen Sie einen Textstring im Speicher ablegen, so wird dazu diese Routine benutzt. In Zeile 720 wird der Text zeichenweise in den Speicher gepoket.

750-760 Diese beiden Programmzeilen ermöglichen es Ihnen, ein Maschinenprogramm im RAM-Speicher auszuführen, dessen Adresse Sie eingeben müssen.

780-810 Zur Auswahl von RAM und ROM dient diese Routine. Der augenblicklich gewählte Zustand wird blinkend angezeigt. Durch Drücken der Leertaste wird von RAM auf ROM und umgekehrt umgeschaltet. Haben Sie den gewünschten Speicher ausgewählt, können Sie durch Drücken der ENTER-Taste wieder zur Befehlseingabe kommen.

830-870 Hier ist das Maschinenprogramm zum Auslesen des ROMs in DATA-Statements abgelegt, die in Zeile 870 an die richtige Stelle im Speicher geschrieben werden.

890-930 Hier geschieht nun das eigentliche Lesen eines Bytes aus dem ROM. Die Adresse des zu lesenden Bytes muß in der Variablen a übergeben werden. Nach der Zerlegung in Hi- und Lo-Byte wird sie in die Maschinenroutine eingefügt. Wenn die Routine mit CALL aufgerufen wird, so hinterlegt sie das Ergebnis in der Speicherzelle &ABOE, aus der sie dann mit PEEK gelesen werden kann.

950 Nach Eingabe von X wird der Bildschirm gelöscht und das Programm beendet.

```
100 MEMORY &AAFF: INK 0,1: INK 1,24: INK 2,23: INK 3,15,24:
    MODE 1
110 n=6 : FOR i=0 TO n: READ cmd$(i): NEXT:
    DATA a,m,s,t,g,r,x
120 GOSUB 830
130 PEN 2: PRINT TAB (10) "M i n i - M o n i t o r"
    CHR$(13)TAB(10);: PEN 1: PRINT CHR$(22)CHR$(1)
    "________________________": PRINT: PRINT
140 PRINT: PRINT "B e f e h l e:";CHR$(13)CHR$(22)CHR$(1)
    "______________"CHR$(22)CHR$(0): PRINT
150 PRINT " m = Speicherbereich anzeigen"
160 PRINT " a = Speicherinhalt aendern"
170 PRINT " s = Speicherbereich abspeichern"
180 PRINT " t = ASCII-Text eingeben"
190 PRINT " g = Maschinenprogramm ausfuehren"
200 PRINT " r = ROM/RAM selektieren"
210 PRINT " x = Rueckkehr zu BASIC"
220 PRINT
230 PRINT: PRINT "Ausgabe auf  ";
240 PEN 3: PRINT "B";: PEN 1: PRINT "ildschirm": PRINT TAB(9)
    "oder ";: PEN 3: PRINT "D";: PEN 1: PRINT "rucker  ?";
250 WHILE a$<>"b" AND a$<>"d": a$=INKEY$:  WEND
260 INK 3,24: rom=0
270 PRINT: IF a$="d" THEN l=16: c=8: GOTO 310
280 PRINT: PRINT: PRINT "Bildschirm-Mode ? "
290 WHILE a$<>"1" AND a$<>"2": a$=INKEY$: WEND
300 a=VAL(a$): l=a*8: c=0: IF a<>1 THEN  MODE a
310 PRINT: PRINT "Befehl ?";
320 a$=INKEY$: FOR i= 0 TO n: IF a$=cmd$(i) THEN PRINT:
    ON i+1 GOSUB 490,540,990,700,750,780,940: GOTO 310
330 NEXT: GOTO 320
340 '
350 IF a>b THEN RETURN
360 PRINT#c, RIGHT$("000"+HEX$(a),4)"  ";
```

```
370 FOR i=1 TO l: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
380 PRINT#c, RIGHT$("0"+HEX$(x),2)+" ";: a=a+1
390 NEXT: PRINT#c, " ";: a=a-l
400 IF c=0 THEN 450
410 FOR i=1 TO l: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
420 x=x AND &7F
430 IF x<32 OR x=127 THEN x=46
440 PRINT#c, CHR$(x);: a=a+1: NEXT: PRINT#c: GOTO 350
450 IF l=8 THEN PEN 2
460 FOR i=1 TO l: IF rom THEN GOSUB 890: ELSE x=PEEK(a)
470 PRINT CHR$(1)CHR$(x);: a=a+1: NEXT: PRINT: PEN 1:
    GOTO 350
480 '
490 INPUT "Adresse ";x$: GOSUB 560: a=x
500 PRINT RIGHT$("000"+HEX$(a),4) ": ";
510 INPUT x$: a$=LEFT$(x$,1): IF a$<"0" OR a$>"9" AND a$<"a"
    OR a$>"f" THEN RETURN
520 GOSUB 560: POKE a,x: a=a+1: GOTO 500
530 '
540 GOSUB 650: GOSUB 350: RETURN
550 '
560 x=VAL("&"+x$): IF x<0 THEN x=x+2^16
570 RETURN
580 '
590 PRINT "Abspeichern:  ";
600 GOSUB 650
610 INPUT "Name ";a$
620 SAVE a$,b,a,b-a
630 RETURN
640 '
650 INPUT "von - bis ";a$,b$
660 x$=a$: GOSUB 560: a=x
670 x$=b$: GOSUB 560: b=x
680 RETURN
```

```
690 '
700 INPUT "Adresse ";x$: GOSUB 560
710 LINE INPUT "Text eingeben: ";a$
720 IF LEN(a$)>0 THEN FOR i=1 TO LEN(a$): POKE x-1+i,
    ASC(MID$(a$,i,1)): NEXT
730 RETURN
740 '
750 INPUT "Adresse ";x$
760 GOSUB 560: CALL x: RETURN
770 '
780 PRINT: INK 3,15,24
790 PRINT "Select: ";: PEN 3: IF rom THEN PRINT "ROM";:
    ELSE PRINT "RAM";
800 PEN 1: a$=INKEY$: IF a$=" " THEN rom=1-rom:
    PRINT CHR$(13);: GOTO 790
810 IF a$=CHR$(13) THEN PRINT: INK 3,24: RETURN: ELSE 800
820 '
830 'Maschinenprogramm laden
840 DATA &DF,&04,&AB,&C9
850 DATA &07,&AB,&FC
860 DATA &3A,&00,&00,&32,&0E,&AB,&C9
870 FOR i=0 TO &D: READ a: POKE &AB00+i,a: NEXT: RETURN
880 '
890 'ROM lesen
900 ah=INT(a/256): al=a-ah*256
910 POKE &AB08,al: POKE &AB09,ah
920 CALL &AB00
930 x=PEEK (&AB0E): RETURN
940 '
950 CLS: END
```

Der Speicher des CPC

Wer sich mit seinem Computer näher beschäftigen und alle Möglichkeiten seines Geräts ausnutzen will, der muß über den internen Aufbau Bescheid wissen.

Der CPC 464 hat als Prozessor einen Z80. Dies ist ein 8-Bit-Prozessor, der 16 Adreßleitungen hat. Über diese 16 Adreßleitungen kann er 2^16 = 65536 Speicherzellen adressieren, die jeweils 8 Bit breit sind und somit 2^8 = 256 unterschiedliche Werte enthalten können. Nun werden Sie vielleicht schon wissen, daß Ihr CPC 464 KByte RAM und 32 KByte ROM enthält. Wie lassen sich diese insgesamt 96 KByte vom Prozessor aus ansprechen, da er doch nur - wie wir oben gesehen haben - 64 KByte adressieren kann?

Dazu bedient man sich eines Tricks. Man belegt den Adreßbereich einfach doppelt, einmal durch das RAM und 'darüber' nochmal durch das ROM. Durch einen 'Schalter' kann der Prozessor nun auswählen, ob er auf das ROM oder das RAM zugreifen will. Beim Schreiben in den Speicher wird automatisch immer das RAM ausgewählt, da ein ROM sich ja nicht beschreiben läßt.

Der CPC 464 enthält einen ROM-Baustein, der die kompletten 32 KByte enthält. Die unteren 16 KByte sind das Betriebssystem, das der Prozessor von Adresse 0 bis &3FFF (16383) anspricht. Die oberen 16 KByte enthalten den BASIC-Interpreter und lassen sich von &C000 bis &FFFF (49152 - 65535) ansprechen. Grafisch läßt sich das folgendermaßen darstellen:

Speicherbelegung

```
&FFFF  -----------   ------------  65535

        BASIC

&C000  -----------                 49152



                        RAM



&4000  -----------                 16384

      BETRIEBS-
       SYSTEM

&0000  -----------   ------------      0
```

Der CPC 464 läßt sich über den Expansionport noch mit weiteren ROM-Modulen erweitern, die jeweils 16 KByte groß sind und parallel zum BASIC-ROM in den Adreßbereich &C000 bis &FFFF gelegt werden, z.B. Anwenderprogramme oder Spiele oder andere Sprachen.

Wenn Sie von BASIC aus den Speicher mit PEEK lesen, so erhalten Sie immer den Inhalt des RAMs zurück. Es ist also nicht so ohne weiteres möglich, Betriebssystem oder BASIC auszulesen. Das später vorgestellte Monitorprogramm bietet jedoch diese Möglichkeit. Doch sehen wir uns jetzt die Aufteilung des RAM-Speichers einmal an.

Die untersten 64 Byte des Speichers sind eine Kopie des ROM-Inhalts des gleichen Adreßbereichs. Wenn Sie den Befehlssatz des Z80-Prozessors kennen, so werden Sie wissen, daß dieser Bereich die Restart-Routinen enthält, die durch einen Ein-Byte-Befehl aufgerufen werden können. Diese acht RST-Vektoren werden beim CPC 464 zum Aufruf von Routinen im ROM und RAM benutzt. Sie übernehmen dabei automatisch die richtige Speicherauswahl. Damit diese Routinen sowohl benutzt werden können, wenn das Betriebssystem-ROM eingeschaltet ist, als auch, wenn das untere RAM selektiert ist, enthält das RAM eine Kopie dieser Routinen. Da die RST-Befehle von Betriebssystem und BASIC ausgiebig genutzt werden, werden wir einige davon kurz besprechen. Eine Anwendung von RST 3 finden wir dann in unserem Monitor, der es uns erlaubt, den ROM-Speicher auszulesen.

Die RST-Befehle

RST 0 Adresse &0000

Über diesen Vektor wird ein kompletter Reset des Computers ausgelöst wie beim Einschalten des Geräts oder beim gleichzeitigen Drücken von CTRL SHIFT ESC.

```
0000  01 89 7F    LD  BC,&7F89
0003  ED 49       OUT (C),C
0005  C3 80 05    JP  &0580
```

Mit dem OUT-Befehl wird das ROM selektiert und dann im Betriebssystem die RESET-Routine ab Adresse &0580 abgearbeitet.

Von BASIC aus können Sie den RESET mit CALL 0, von Maschinensprache aus mit CALL 0, JP 0 oder RST 0 aufrufen.

RST 1 Adresse &0008

Dieser Befehl dient zum Aufruf einer Routine im Betriebssystem oder im darunterliegenden RAM. Direkt hinter dem RST-Befehl muß die Adresse der aufzurufenden Routine stehen. Da für den Bereich von 0 bis &3FFF 14 Adreßbits ausreichen, benutzt man die oberen beiden Bits für die Auswahl von ROM oder RAM:

```
Bit 14 = 0    Betriebssystem ausgewählt
       = 1    RAM ausgewählt
Bit 15 = 0    BASIC-ROM ausgewählt
       = 1    RAM ausgewählt
```

Ein Aufruf der Betriebssystemroutine &0826 könnte dann so aussehen:

```
RST 1
DW  &0826 + &8000
```

Durch das gesetzte Bit 15 ist im Bereich von &C000 bis &FFFF RAM selektiert, während durch das gelöschte Bit 14 das Betriebssystem angesprochen wird.

Der Kode an der Adresse 10 besteht lediglich aus einem Sprung zu &B982.

RST 2 Adresse &0010

Dieser Restart-Befehl dient zum Aufruf einer Routine in einem Expansion-ROM. Nach dem RST 2-Befehl muß die Adresse der Routine - &C000, d.h. also die relative Adresse bezogen auf den Start des ROMs, stehen. Die obersten beiden Bits werden zur Auswahl von vier verschiedenen ROMs benutzt. An Adresse &0010 steht ein Sprung zu &BA16.

RST 3 Adresse &0018

Mit Hilfe dieses RST-Befehls können Sie eine Routine irgendwo im ROM oder RAM aufrufen. Dazu muß hinter dem RST 3-Befehl die Adresse eines Parameterblocks stehen, der aus drei Bytes besteht. Diese ersten beiden Bytes enthalten die Adresse der Routine, die aufgerufen werden soll, und das dritte Byte muß den gewünschten ROM/RAM-Status enthalten. Dabei wird durch die Werte von 0 bis 251 das entsprechende Zusatzrom angesprochen. Die verbleibenden vier Werte haben folgende

Funktion:

Wert	&0000-&3FFF	&C000-&FFFF
252	Betriebssystem	BASIC
253	Betriebssystem	RAM
254	RAM	BASIC
255	RAM	RAM

An Adresse &0018 steht ein Sprung nach &B9BF.

RST 4 Adresse &0020

Mit Hilfe dieses RST-Befehls können Sie von einem Maschinenprogramm den Inhalt des RAMs lesen, unabhängig vom jeweils gewählten ROM-Zustand. Der RST 4-Befehl ersetzt dabei den Befehl

```
LD  A,(HL)
```

HL muß dazu also die Adresse der zu lesenden Speicherzelle enthalten. An Adresse &0020 steht ein Sprung zu &BACB.

RST 5 Adresse &0028

Mittels dieses RST-Befehls kann man zu einer Routine im Betriebssystem springen. Die Adresse muß dabei unmittelbar auf den RST 5-Befehl folgen. An Adresse &0028 steht ein Sprung zu &BA2E.

RST 6 Adresse &0030

Dieser RST-Befehl wird vom Betriebssystem nicht benutzt und steht dem Benutzer zur freien Verfügung und kann z.B. in einem Debugger benutzt werden, um einen Breakpoint zu setzen.

RST 7   Adresse &0038

Da der Z80 im CPC 464 im Interrupt-Mode 1 betrieben wird, wird beim Auftreten eines Interrupts ein RST 7 ausgelöst. Über diesen Befehl wird der Interrupt behandelt. An Adresse &0038 steht ein Sprung zu &B939.

Datenübertragung zu anderen Computern

Wollen Sie Daten vom Ihrem CPC 464 zu einem anderen Computer übertragen, so ist dies mittels Band oder Diskette meist nicht möglich, da die Aufzeichnungsformate nicht kompatibel sind. Ohne den Umweg über einen Massenspeicher ist dies jedoch mit den meisten Computern möglich. Der Rechner, auf den Sie die Daten vom CPC übertragen wollen, muß dazu nur eine Parallelschnittstelle besitzen, die als Eingang benutzt wird. Sämtliche Commodore-Rechner verfügen z.B. über eine solche Schnittstelle.

Ihr CPC hat ebenfalls eine Parallelschnittstelle, die sich jedoch nur zur Datenausgabe benutzen läßt. Es ist die Druckerschnittstelle, an die normalerweise ein Drucker mit Centronicsschnittstelle angeschlossen wird. Das Übertragungsprotokoll einer solchen Schnittstelle ist besonders einfach und läßt sich auf dem Empfangsrechner durch ein kleines Programm realisieren. Um die beiden Rechner zu verbinden, ist lediglich ein Kabel erforderlich, das den Printer-Anschluß des CPCs mit dem Userport z.B. eines Commodore verbindet. Da der Drucker vom CPC bereits unterstützt wird, kann man diese Schnittstelle einfach über die Kanalnummer 8 ansprechen. Diese Methode wurde z.B. benutzt, um die in diesem Buch abgedruckten Programmlistings in die Textverarbeitung eines Commodore-Rechners zu übernehmen. Da, wie gesagt, die Schnittstelle vom CPC schon unterstützt wird, geben wir Ihnen hier lediglich als Beispiel ein kleines Maschinenprogramm für den Commodore 64 an. Es übernimmt die Daten vom CPC, wandelt sie in den Commodore-Code wandelt und schreibt sie in eine sequentielle Datei z.B. auf Kassette, Diskette oder einfach auf den Bildschirm. Wenn Sie dieses Programm auf einem C64 aktiviert haben, können Sie vom CPC aus einfach mit

LIST #8

ein Programm auf den C64 übertragen. Sonstige beliebige Texte oder Zeichen können Sie mit einem PRINT #8 Befehl übertragen, z.B.

PRINT #8, "Eine Nachricht vom CPC 464"

Das Maschinenprogramm auf dem C64 wurde so ausgelegt, daß die Übertragung vom CPC 464 durch Senden von CHR$(0) beendet werden kann.

Dabei ist noch eine Besonderheit des CPC 464 zu beachten. Eine Centronicschnittstelle arbeitet normalerweise mit 8 Datenbits. Der CPC 464 stellt jedoch nur 7 Bits an der Schnittstelle zur Verfügung, das oberste Datenbit ist permanent auf Masse gelegt. Bei der Übertragung von ASCII-Texten ist dies nicht weiter störend, da der ASCII-Kode ein 7-Bit-Kode ist. Anders wird es da schon, wenn Sie z.B. eine Hardcopy vom Grafikbildschirm auf den Drucker machen. Die meisten Drucker arbeiten mit einer 8-Nadel-Grafik, so daß eine Grafikhardcopy mit Problemen verbunden sein wird. Doch zurück zur Rechnerübertragung. Hier nun das Maschinenprogramm für den C64.

```
30:   DD00              CIA    =    $DD00
40:   DD00              PORTA  =    CIA        ; Port Für 'Busy'
50:   DD01              PORTB  =    CIA+1      ; Port für Daten
60:   DD0D              CR     =    CIA+13     ; Controll Register
70:                     ;
80:   C000                     *=   $C000
90:   C000 AD 00 DD START      LDA  PORTA
100:  C003 29 FB               AND  #%11111011  ; Busy low
```

```
110:   C005 8D 00 DD          STA  PORTA
120:   C008 A9 10             LDA  #%10000
130:   C00A 2C 0D DD WAIT     BIT  CR
140:   C00D F0 FB             BEQ  WAIT     ; Warten auf Strobe
150:   C00F AD 00 DD          LDA  PORTA
160:   C012 09 04             ORA  #%100    ; Busy hi
170:   C014 8D 00 DD          STA  PORTA
180:   C017 AD 01 DD          LDA  PORTB    ; Daten lesen
180:   C01A F0 2A             BEQ  ENDE     ; Null, dann Ende
180:   C01C C9 0A             CMP  #10      ; Line feed ignorieren
180:   C01E F0 E0             BEQ  START
181:   C020 C9 41             CMP  #"A"
181:   C022 90 12             BCC  END      ; Umwandlung ins
182:   C024 C9 5B             CMP  #"Z"+1
182:   C026 B0 04             BCS  TEST     ; CBM-ASCII
183:   C028 09 80             ORA  #$80
183:   C02A 30 0A             BMI  END
184:   C02C C9 61    TEST     CMP  #$61
184:   C02E 90 06             BCC  END
185:   C030 C9 7B             CMP  #$7B
185:   C032 B0 02             BCS  END
186:   C034 29 5F             AND  #$5F
189:   C036 48       END      PHA           ; Daten merken
189:   C037 A2 01             LDX  #1
189:   C039 20 C9 FF          JSR  $FFC9    ; Ausgabe auf Kanal 1
190:   C03C 68                PLA
190:   C03D 20 D2 FF          JSR  $FFD2    ; Zeichen ausgeben
195:   C040 20 CC FF          JSR  $FFCC    ; Ausgabe wieder auf Default
200:   C043 4C 00 C0          JMP  START    ; und zum Start
210:   C046 60       ENDE     RTS           ; fertig
```

Vom Commodore 64 aus wird zuerst ein Datenkanal mit der logischen Nummer eins geöffnet, z.B.

```
OPEN 1,1,1,"DATEN" für Kassette oder

OPEN 1,8,2,"DATEN,S,W" für Diskette bzw.

OPEN 1,3
```

wenn nur auf den Bildschirm ausgegeben werden soll. Dann kann das Programm mit SYS 49152 gestartet werden. Nun können Sie z.B. mit LIST #8 ein Programmlisting vom CPC zum C64 schicken. Beenden können Sie die Übertragung, indem Sie

```
PRINT #8, CHR$(0);
```

auf dem CPC 464 eingeben. Der C64 meldet sich dann mit ´READY.´ wieder und Sie können mit CLOSE 1 die Datei, in die die Daten geschrieben wurden, schließen. Zum Abschluß noch die Pinbelegung für das Verbindungkabel zwischen den beiden Rechnern.

CPC 464 Drucker-Port Pin	Bedeutung	C64 User-Port Pin
1	STROBE	B
2	D0	C
3	D1	D
4	D2	E
5	D3	F
6	D4	H
7	D5	J
8	D6	K
9	D7	L
11	BUSY	M
12	GND	N

Sortieren von Daten

Eine Aufgabe, auf die jeder Programmierer früher oder später stößt, ist das Sortieren von Daten. Dabei kann es sich um numerische Daten handlen oder um Texte, z.B. die Datensätze einer Adreßdatei.

Als Ordnungskriterium zum Sortieren kann man bei Zahlen einfach ihre Größe nehmen; Texte werden aufgrund des ASCII-Kodes sortiert, in dem sie kodiert sind. Der Kode ist so gewählt, daß z.B. ´A´ kleiner als ´B´ ist und dieses wiederum kleiner als ´C´. Vor den Großbuchstaben sind noch einige Sonderzeichen sowie die Ziffern angeordnet, die Kleinbuchstaben folgen auf die Großbuchstaben. Wenn wir eine Sortierroutine schreiben, so brauchen wir uns darüber weiter nicht zu kümmern, den der BASIC-Interpreter erlaubt es direkt, zwei Strings miteinander zu vergleichen und berücksichtigt dabei die obigen Kriterien.

Will man Daten sortieren, so kann man aus einer Vielzahl bekannter Algorithmen auswählen. Diese Algorithmen unterscheiden sich zum einen durch ihre Komplexität und zum anderen durch ihre Effizienz. Bei der Auswahl eines geeigneten Verfahrens spielt die Anzahl der zu sortierenden Daten eine wichtige Rolle. Haben so bloß 10 oder 50 Datensätze zu sortieren, so reichen meist die einfachsten Verfahren aus. Sollen dagegegen mehrere hundert oder gar tausend Sätze sortiert werden. so sind die simplen Verfahren nicht mehr brauchbar, da sich die Sortierzeit dann über mehrere Stunden hinziehen wird. Um Ihnen einen Eindruck davon zu vermitteln, wollen wir das einfachste und das schnellste Verfahren vorstellen, Bubble-Sort und Quick-Sort.

Beim Bubble-Sort werden jeweils zwei benachbarte Elemente verglichen und falls erforderlich ausgetauscht. In BASIC läßt sich das mit zwei verschachtelten Schleifen programmieren.

```
100 FOR i=1 TO N: fl=0
110 FOR j=n TO i step -1
120 IF a$(j-1)>a$(j) THEN h$=a$(j):a$(j)=a$(j-1):a$(j-1)=h$:
                          fl=1
130 NEXT j: IF fl=0 THEN RETURN
140 NEXT i: RETURN
```

Das Programm wird mit GOSUB 100 als Unterprogramm aufgerufen. Die zu sortierenden Daten müssen als Stringarray in a$ stehen. Die Größe des Feldes (der maximale Index) muß in der Variablen n übergeben werden, z.B.

```
10 n=50: DIM a$(n)
20 ´Einlesen der Daten
30 GOSUB 100 ´Daten sortien
```

Ein solches einfaches Sortierprogramm eignet sich immer dann, wenn nur wenige Daten sortiert werden müssen oder wenn die Daten schon vorsortiert sind und z.B. nur ein neuer Datensatz einsortiert werden soll.

Das nun vorgestellte Sortierprogramm, daß unter dem Namen Quick-Sort bekannt ist, ist das geeignete Verfahren wenn Sie größere Datenmengen zu sortieren haben. Hier wird ebenfalls ein Stringfeld sortiert, dessen Indexgrenze in n enthalten sein muß.
Das Programm generiert zu Demonstration eine von Ihnen anzugebende Anzahl von Zufallsworten, zeigt diese an, sortiert sie und zeigt Sie in der sortieren Reihenfolge nocheinmal an zusammen mit der Zeit, die dazu benötigt wurde.

```
10 DEFINT b-z
20 INPUT "Anzahl der Worte ";n: DIM d$(n),o(20),u(20)
30 FOR i=1 TO n: b=5+10*RND
40 FOR j=1 TO b: d$(i)=d$(i)+CHR$(RND*25+65):NEXT: NEXT
50 GOSUB 80: a=TIME: GOSUB 100: a=(TIME-a)/300
60 PRINT: GOSUB 80: PRINT: PRINT a "Sek.": END
70 '
80 FOR i=1 TO n: PRINT d$(i): NEXT: RETURN
90 '
100 s=1: o(1)=1: u(1)=n
110 l=o(s): r=u(s): s=s-1
120 i=l: j=r: h$=d$((l+r)/2)
130 WHILE d$(i)<h$ AND i<r: i=i+1: WEND
140 WHILE d$(j)>h$ AND j>l: j=j-1: WEND
150 IF i<=j THEN d$=d$(i):d$(i)=d$(j):d$(j)=d$:i=i+1:j=j-1
160 IF i<=j THEN 130
170 IF r-i<=j-l THEN 200
180 IF l<j THEN s=s+1: o(s)=l: u(s)=j
190 l=i: GOTO 220
200 IF i<r THEN s=s+1: o(s)=i: u(s)=r
210 r=j
220 IF r>l THEN 120
230 IF s>0 THEN 110
240 RETURN
```

Wenn Sie das Programm einmal mit verschieden großen Feldern laufen lassen, werden Sie etwa folgende Ergebnisse erhalten:

n	Sek.
50	3.8
100	7.9
200	18.6
500	110

Das die Zeit für 500 Datensätze unverhältnismäßig hoch ist, hat mit einem anderen Phänomen zu tun. Es handelt sich dabei um die sogenannte Garbage Collection, die immer dann in Aktion tritt, wenn mit vielen Strings gearbeitet wird. Wenn Sie das letzte Beispiel mit den 500 Strings gerade ausgeführt haben, so geben Sie einmal

```
? FRE("")
```

ein - Sie werden ca. 29 Sekunden warten müssen. Diese Zeit benötigt der BASIC-Inter, um seinen Stringspeicher aufzuräumen. Jedesmal wenn der Inhalt einer Stringvariablen verändert wird, schreibt der Interpreter den neuen Inhalt in den freien Speicher während der alte Stringinhalt weiterhin im Speicher stehen bleibt. Das geht solange gut, bis der komplette Speicher mit Strings vollgeschrieben ist. Der größte Teil dieser Strings wird jedoch nicht mehr benötigt und kann gelöscht werden. Dazu wird der komplette Speicher durchsucht und alle Strings, die nicht mehr benötigt werden, werden gelöscht bzw. mit den gültigen Strings überschrieben, so daß wieder Platz zum Anlegen neuer Strings zur Verfügung steht. Je mehr gültige Strings nun im Speicher stehen, desto mehr Zeit benötigt dieses Vefahren, bei 500 Strings eben diese 29 Sekunden. Wenn als beim Sortieren, wo ständig Strings umkopiert werden müssen, der Speicherplatz nicht mehr ausreichen sollte, muß eine Garbage Collection eingefügt werden, die die Sortierzeit unverhältnismäßig hoch ansteigen läßt. Zum Vergleich noch die Zeiten für das Bubble-Sort-Verfahren.

n	Sek.
50	12.1
100	49.0
200	238.2

Deutsche Umlaute für den CPC

Wollen Sie Ihren CPC für Textverarbeitung oder eine Dateiverwaltung einsetzen, so wird man meist eine Möglichkeit suchen, auch die deutschen Umlaute darzustellen, die der Rechner von Hause aus nicht beherrscht.

Da der CPC jedoch sowohl die Darstellung jeden Zeichens auf dem Bildschirm frei definieren kann als auch jeder Taste einen beliebigen Kode zuordnen kann, haben wir damit schon alle Voraussetzungen dafür zusammen.

Zwei Befehle sind dafür zuständig: Mit SYMBOL können wir einem ASCII-Zeichen auf dem Bildschirm ein beliebiges Aussehen verleihen. Wie Sie sicher wissen, wir jedes Zeichen aus einer 8*8 Punktmatrix zusammengesetzt. Wollen wir nun ein neues Zeichen definieren, so zeichnen wir uns eine solche Matrix auf und markieren die gesetzten Punkte. Wenn wir die binäre Darstellungsweise wählen, können wir das Bitmuster direkt übernehmen. Sehen wir uns das am Beispiel des großen Ä einmal an.

```
   12345678
1  01011010
2  00111100
3  01100110
4  01100110
5  01111110
6  01100110
7  01100110
8  00000000
```

Dabei bedeutet eine 1 einen gesetzen Punkt und eine 0 einen nicht gesetzten Punkt. Ehe wir jedoch an die Definition gehen, müssen wir uns darüber klar werden, welche ASCII-Kodes wir für die Umlaute reservieren. Dazu gibt es keine Norm; die üblichste Zuordnung besteht darin, die großen Umlaute hinter dem ´Z´ anzuordnen um die kleinen Umlaute einschließlich ´ßß´ hinter dem ´z´. Es ergibt sich dann folgende Zuordnung der Umlaute zu den ASCII-Kodes:

Ä	91	&5B
Ö	92	&5C
Ü	93	&5D
ä	123	&7B
ö	124	&7C
ü	125	&7D
ß	126	&7E

Wenn wir diese Kodierung wählen, werden die Umlaute auf einem angeschlossenen Epson-Drucker ebenfalls korrekt wieder gegeben. Normalerweise sind diese ASCII-Kodes durch die eckigen und geschweiften Klammern bzw. den Backslash belegt. Wir definieren dazu die Tastenbelegungen so um, daß ´eckige Klammer auf´ dem ä entspricht, der ´Backslash´ (über der CTRL-Taste) dem ö und die ´eckige Klammer´ zu dem ü. Das ß legen wir auf den ´Klammeraffen´. Mit diesen Vorgaben können wir nun folgendes kleine Programm schreiben, daß die Umdefinitionen vornimmt.

```
100 REM DEFINITION DER DEUTSCHEN UMLAUTE
        UND DER TASTENBELEGUNGEN
110 SYMBOL AFTER 90
120 SYMBOL 91, &X01011010,&X00111100,&X01100110,&X01100110,
               &X01111110,&X01100110,&X01100110,&X00000000
```

```
130 SYMBOL 92, &X10111010,&X01101100,&X11000110,&X11000110,
               &X11000110,&X01101100,&X00111000,&X00000000
140 SYMBOL 93, &X01100110,&X00000000,&X01100110,&X01100110,
               &X01100110,&X01100110,&X00111100,&X00000000
150 SYMBOL 123,&X01001000,&X00000000,&X01111000,&X00001100,
               &X01111100,&X11001100,&X01110110,&X00000000
160 SYMBOL 124,&X00100100,&X00000000,&X00111100,&X01100110,
               &X01100110,&X01100110,&X00111100,&X00000000
170 SYMBOL 125,&X01000100,&X00000000,&X01100110,&X01100110,
               &X01100110,&X01100110,&X00111110,&X00000000
180 SYMBOL 126,&X00111000,&X01101100,&X01101100,&X01101100,
               &X01100110,&X01110110,&X01101100,&X01100000
190 KEY DEF 22,1,124,92
200 KEY DEF 19,1,125,93
210 KEY DEF 17,1,123,91
220 KEY DEF 26,1,126,96
```

Beachten Sie bei der Eingabe, daß der Editor führende Nullen unterdrückt, d.h &X00110011 wird als &X110011 gelistet. Wenn Sie dieses Programm an den Anfang Ihres Hauptprogramm stellen oder als Unterprogramm aufrufen, können Sie fortan den deutschen Zeichensatz benutzen.

```
0160  00 00 00 00 00 00 00 00   ........
0168  00 00 00 00 00 00 00 00   ........
0170  31 00 64 00 AA 20 1C FF   1.d.* ..
0178  AA 01 20 A2 20 0E 2C 0F   *. " .,.
0180  01 20 A2 20 0F 2C 19 18   . " .,..
0188  01 20 A2 20 10 2C 19 17   . " .,..
0190  01 20 A2 20 11 2C 19 0F   . " .,..
0198  2C 19 18 01 20 AD 20 0F   ,... - .
01A0  00 41 00 6E 00 0D 05 00   .A.n....
01A8  EE EF 14 20 01 20 9E 20   no. . .
01B0  0D 0E 00 E9 EF 0E 20 EC   ...io. l
01B8  20 0D 05 00 EE 01 20 C3    ...n. C
01C0  20 03 09 00 63 6D E4 28    ...cmd(
01C8  0D 0E 00 E9 29 01 20 B0   ...i). 0
01D0  01 20 8C 20 61 2C 6D 2C   . . a,m,
01D8  73 2C 74 2C 67 2C 72 2C   s,t,g,r,
01E0  78 00 0A 00 78 00 9F 20   x...x..
01E8  1D 06 0B 00 6B 00 82 00   ....k...
01F0  BB 20 10 01 20 BF 20 EA   ; .. ? j
01F8  20 28 19 0A 29 20 22 4D    (..) "M
0200  20 69 20 6E 20 69 20 2D    i n i -


C000  80 01 00 00 4C C0 31 00   ....L@1.
C008  C0 CD CB BC CD C4 F4 DA    MK<MDtZ
C010  00 00 21 00 AC 36 00 06   ..!.,6..
C018  1B 23 36 C9 10 FB 21 3F   .#6I.;!?
C020  C0 CD 37 C3 AF 32 00 AC   @M7C/2.,
C028  CD CB DD CD 84 CA CD 97   MK M.JM.
C030  BD CD D3 C0 CD 3E C1 11   =MS@M>A.
C038  F0 00 CD 06 F7 18 25 20   p.M.w.%
C040  42 41 53 49 43 20 31 2E   BASIC 1.
C048  30 0A 0A 00 42 41 53 49   0...BASI
C050  C3 00 CD E1 CE C0 31 00   C.MaN@1.
```

```
C058   C0 CD 9A E7 CD 63 E1 CD   @M.gMcaM
C060   43 CA 38 54 CD 01 AC 31   CJ8TM.,1
C068   00 C0 CD 62 C1 CD D6 DD   .@MbAMV
C070   DC B6 BC CD 48 BB CD 86    6<MH;M.
C078   C3 3A 45 AE B7 C4 3E C1   C:E.7D>A
C080   3A AA AD D6 02 20 09 32   :*-V. .2
C088   AA AD CD DF CA EB 38 C6   *-M
C090   21 CC C0 CD 41 C3 CD CB   !L@MACMK
C098   DD 3A 1C AC B7 28 11 CD    :.,7(.M
C0A0   02 C1 30 C0 7E B7 28 F1   .AO@>7(q
C0A8   CD D2 E6 CD 7A C1 18 E9   MRfMzA.i
C0B0   CD 3B CA 30 FB CD 4E C3   M;JO;MNC
C0B8   CD BC E6 30 05 C4 7A C1   M<fO.DzA
C0C0   18 D4 CD BB DE CD 53 C4   .TM;^MSD
C0C8   2B C3 74 DD 52 65 61 64   +Ct Read
C0D0   79 0A 00 AF 18 05 22 1D   y../..".
C0D8   AC 3E FF 32 1C AC C9 11   ,>.2.,I.
C0E0   0A 00 28 02 FE 2C C4 E1   ..(.>,Da
C0E8   CE D5 11 0A 00 CD 55 DD   NU...MU
C0F0   DC E1 CE CD 4A DD EB 22    aNMJ k"
C0F8   1F AC E1 CD D6 C0 C1 C3   .,aMV@AC
```

K A P I T E L 5 BASIC-ZEILEN, VARIABLE UND TOKENS

5.1 ABLAGE EINER BASICZEILE

Zum Nachvollziehen dieses Kapitels benötigen Sie das Maschinensprachemonitorprogramm aus Kapitel 4. Sollten Sie es noch nicht abgetippt haben, so raten wir Ihnen, es möglichst noch zu tun, damit keine Verständnisschwierigkeiten auftreten.

In diesem Kapitel wird ein Thema angeschnitten, das man als Bindeglied zwischen Maschinensprache und Basic bezeichnen könnte. Es geht um die Ablage einer Basiczeile und deren Aufbau. Weiterhin wollen wir Ihnen die Ablage von Variablen in einer Basiczeile verdeutlichen.

Wie Sie aus Kapitel 4 wissen, versteht Ihr CPC 464 nur den Binärcode. Durch das Betriebssystem und den Basicinterpreter (Übersetzer) werden alle Basicbefehle in diesen Binärcode umgewandelt. Schreiben Sie also die Basiczeile 1 PRINT "DATA BECKER", so versteht Ihr CPC ohne Basicinterpreter nichts. Jedoch versteht der Rechner mit Interpreter auch nicht alles. Das haben Sie sicher am Anfang Ihrer Programmierversuche an dem vielen "SYNTAX ERROR"-Meldungen erkannt.

Nun aber zum praktischen Teil: Ergänzen Sie Ihr Maschinensprachemonitorprogramm mit der oben schon angegebenen Zeile 1 PRINT "DATA BECKER". Starten Sie nun das Monitorprogramm und wählen das Dezimalsystem und den Lesemodus. Als Anfangsadresse geben Sie bitte 368 und als Endadresse 386 ein. Den folgenden Ausdruck haben wir hier noch einmal einzeln untereinander aufgeschreiben, damit keine Mißverständnisse entstehen. Wir haben in der rechten Spalte auch noch einen kurzen Kommentar hinzugefügt, jedoch wird alles anschließend noch ausführlich erklärt.

ADRESSE:	WERT:	KOMMENTAR:
368	20	Lowbyte der Zeilenlänge
369	0	Highbyte der Zeilenlänge
370	1	Lowbyte der Zeilennummer
371	0	Highbyte der Zeilennummer
372	191	TOKEN für PRINT
373	32	ASCII Wert für Leerstelle
374	34	ASCII Wert für Anführungszeichen
375	68	ASCII Wert für "D"
376	65	ASCII Wert für "A"
377	84	ASCII Wert für "T"
378	65	ASCII Wert für "A"
379	32	ASCII Wert für Leerzeichen
380	66	ASCII Wert für "B"
381	69	ASCII Wert für "E"
382	67	ASCII Wert für "C"
383	75	ASCII Wert für "K"
384	69	ASCII Wert für "E"
385	82	ASCII Wert für "R"
386	34	ASCII Wert für Anführungszeichen

Wir wollen die Werte von oben nach unten erklären. Die Begriffe Low- und Highbyte dürften Ihnen aus Kapitel 2 noch in Erinnerung sein, wenn nicht, so hier noch einmal die Formel zur Berechnung der eigentlichen Zahl, die durch Highbyte und Lowbyte ausgedrückt wird:

HIGHBYTE*256+LOWBYTE

Es ergibt sich als Zeilenlänge also die Zahl 20. Wenn Sie die oben abgedruckten Adressen zählen, werden Sie nur auf die Zahl 19 kommen. Dies liegt daran, daß für Ihren CPC die

eigentliche Zeilenlänge uninteressant ist, denn er benötigt diesen Wert nur, um die Anfangsadresse der nächsten Zeile zu errechnen. Die Zeilennummer besteht auch wieder aus High- und Lowbyte, wenn es nicht so wäre, gäbe es nur Zeilennummern bis 255. Die Zeilennummer ist 1 und das können Sie auch aus High- und Lowbyte ersehen. Der nächste Wert ist ein sogenannter Token. Dieses Wort und seine Bedeutung wird im folgenden Kapitel 5.2 ausführlich erklärt, denn es ist einer der wichtigsten Begriffe im Zusammenhang mit Basiczeilenaufbau.

Die folgenden Werte sind die ASCII Werte für den Schriftzug "DATA BECKER".

So wie diese Zeile aufgebaut ist, so ist im Prinzip jede Zeile des Monitorprogramms oder jedes anderen Programms aufgebaut. Es gibt nur eine krasse Ausnahme, und das ist eine Basiczeile, in der eine Variable definiert wird. Auf diese Ausnahme kommen wir später noch zurück.

5.2 TOKENS

Der Begriff TOKEN kommt, wie fast jedes Fachwort in der Computertechnik, aus dem Englischen und bedeutet Zeichen. Der Token ist ein Wert, der einem Basicbefehl zugeordnet wird. Genau wie die Binärwerte den Maschinensprachebefehlen. Er wurde entwickelt, um Speicherplatz zu sparen. Der Token für PRint ist beispielsweise nur ein Byte lang. Wenn Sie jetzt die Buchstaben zählen, kommen Sie auf fünf Werte, das heißt, normalerweise würden also für PRINT fünf Bytes benötigt. Weiterhin kann der Rechner den Befehl besser von irgendwelchen Variablen mit einem ähnlichen Namen unterscheiden. Darum ist es auch nicht zulässig, eine Variable mit dem Namen PRINT zu definieren, da der Computer diese Buchstabenzusammenstellung als Befehl interpretiert und einen Token einsetzt.

Wir haben Ihnen eine Liste aller Tokens zusammengestellt, anhand derer Sie alle Monitorausdrücke eines Basicprogramms interpretieren können.
Noch ein guter Tip zum Programmschutz:
Schreiben Sie als erste Zeile Ihres Programms eine REM Zeile, in der Sie mit Anführungsstrichen Ihr Copyright einsetzen. Das müßte dann ungefähr so aussehen:

```
1 REM "COPYRIGHT BY DATA BECKER"
```

Setzen Sie in die zweite Zeile eine wichtige Anweisung, ohne die das Programm nicht läuft. Eine solche Anweisung wäre zum Beispiel ein DIM oder eine Funktionsdefinierung. Setzen Sie nun mitten im Programm eine Zeile mit folgenden Inhalt ein.

```
POKE 372,191
```

Dies bewirkt, daß aus der ersten REM Zeile eine PRINT Zeile wird. Fehlt die REM Zeile, so wird die zweite Zeile mit der wichtigen Anweisung zerstört. Dieser kleine Trick verhindert, daß Unbefugte die COPYRIGHT Zeile herausnehmen.

Befehl	Wert	Befehl	Wert	Befehl	Wert
Auto	129	Mode	173	Wait	212
Border	130	Move	174	Wend	213
Call	131	Mover	175	While	214
Cat	132	Next	176	Width	215
Chain	133	New	177	Window	216
Chain Merge	133&171	On	178	Write	217
Clear	134	On Break	179	Zone	218
Clg	135	On Error goto	180	Di	219
Close in	136	On sq	181	Ei	220
Close out	137	Open in	182	Erl	227
Cls	138	Open out	183	Fn	228
Cont	139	Origin	184	Spc	229
Data	140	Out	185	Step	230
Def fn	141	Paper	186	Swap	231
Def int	142	Pen	187	Tab	234
Def read	143	Plot	188	Then	235
Def Str$	144	Plotr	189	To	236
Deg	145	Poke	190	Using	237
Delete	146	Print	191	Chr$(62)	238
Dim	147	Chr$(39)	192	Chr$(61)	239
Draw	148	Rad	193	Chr$(62)+"="	240
Drawr	149	Randomize	194	Chr$(60)	241
Edit	150	Read	195	Chr$(60)+(62)	242
Else	151	Release	196	Chr$(60)+(61)	243
End	152	Rem	197	Chr$(43)	244
Ent	153	Renum	198	Chr$(45)	245
Env	154	Restore	199	Chr$(42)	246
Erase	155	Resume	200	Chr$(47)	247
Error	156	Return	201	Chr$(94)	248
Every	157	Run	202	Chr$(92)	249
For	158	Save	203	And	250
Gosub	159	Sound	204	Mod	251
Goto	160	Speed	205	Or	252

If	161	Stop	206	Xor	253
Ink	162	Symbol	207	Not	254
Input	163	Tag	208	Abs	255
Key	164	Tag off	209		
Key Def	164&141	Troff	210		
Let	165	Tron	211		
Line Input	166&163				
List	167				
Load	168				
Locate	169				
Memory	170				
Merge	171				
Mid	172				

Die Tokenwerte 221 bis 226 sind nicht benutzt, erzeugen aber eine recht interessante Wirkung. Schreiben Sie als erste Zeile 1 REM, und merken Sie sich die zweite Zeilennummer. Schreiben Sie nun POKE 372,226. Wenn Sie danach versuchen, das Programm mit LIST zu listen, erscheint ein "SYNTAX ERROR". Wenn Sie das Programm nur mit "RUN" starten, zeigt Ihr CPC dauernd einen "SYNTAX ERROR IN 1" an, und Sie können dies nur durch ein RESET (gleichzeitiges Drücken der Tasten "CTRL", "SHIFT" und "ESC") unterbrechen, jedoch geht dabei Ihr Programm verloren. Starten Sie es deshalb mit GOTO, zweite Zeilennummer, die Sie sich hoffentlich gemerkt haben. Ein nicht Eingeweihter wird das Programm mit "RUN" starten und es dadurch zerstören. Wollen Sie diesen Programmschutz wieder aufheben, so schreiben Sie POKE 372,197.
Noch eine kurze Anmerkung: Ihr Programm, das diesen Programmschutz trägt, darf kein GOTO 1 oder GOSUB 1 beinhalten, da es sonst auch zerstört wird.

5.3 ABLAGE VON VARIABLEN

Wie schon in Kapitel 5.1 angesprochen, ist die große Ausnahme unter den Basiczeilen die Zeile, in der eine Variable definiert wird. Wir werden versuchen, Ihnen dieses äußerst problematische Thema so einfach wie möglich darzulegen. Dafür haben wir uns fünf repräsentative Fälle von Variablen ausgesucht.

Bevor Sie weiterlesen, möchten wir Sie bitten, da Maschinensprachemonitorprogramm aus Kapitel 2 zu laden, damit Sie unseren Ausführungen auch vollständig folgen können.

Geben Sie nun als erstes Beispiel die Zeile 1 C=100 ein.

Starten Sie das Monitorprogramm, wählen Sie wieder das Dezimalsystem und den Lesemodus, als Anfangsadresse geben Sie bitte 368 und als Endadresse 379 ein.

Wir haben den Monitorausdruck aufgeführt und mit kurzen Kommentar versehen:

Adresse	Wert	Kommentar
368	12	Lowbyte der Zeilenlänge
369	0	Highbyte der Zeilenlänge
370	1	Lowbyte der Zeilennummer
371	0	Highbyte der Zeilennummer
372	13	Zeigt an, daß Zahlvariable
373	5	Variablennamenlänge +4
374	0	Trennungsnull
375	227	ASCII Wert des Variablennamens +128
376	239	Tokenwert für "="
377	25	Variablengröße
378	100	Wert der Variable
379	0	Trennungsnull

Die Erklärung zu den Werten der ersten vier Adressen finden Sie in Kapitel 5.1. Nun zum Wert 13 in Adresse 372. Da steht normalerweise der Token für den Befehl. Hier hingegen nicht, es wird angezeigt, daß die Variable C eine numerische Variable ist. Hätten wir eine Stringvariable verwendet, würde dort eine "3" stehen.

Die Variablennamenlänge in Adresse 373 ist folgendermaßen verschlüsselt:

Wert = Variablennamenlänge + 4. Die folgende Null steht nur als Trennung dort und hat keine weitere Bedeutung, ebenfalls die Null in 379. Nun zum Variablennamen. Er wird auf folgende Weise aufgeschlüsselt: Zum ASCII Wert des letzten Buchstabens des Variablennamens wird die Zahl 128 addiert. Alle anderen Buchstaben des Variablennamens werden mit ihrem ASCII Wert abgelegt. Der Wert 239 in Adresse 376 ist der Tokenwert für das Gleichheitszeichen. Man hat hier nicht den ASCII Wert genommen, damit es klar wird, daß das "=" nicht zum Variablennamen gehört. An dieser Stelle ist der Variablenname zu Ende ist. Der nächste Wert (25) ist der verschlüsselte Wert für die Variablengröße. Er sagt dem Rechner, daß der Wert der Variablen in ein Byte paßt, also nicht größer als 255 ist, sowie keine Nachkommastellen besitzt. Der Wert der Variablen ist eine Integerzahl (ganze Zahl). Die verschiedenen Werte für die Variablengröße haben wir Ihnen später auch noch in einer Tabelle zusammengestellt. Jetzt zur "100" in Adresse 378: Dies ist der Wert der Variablen, denn wir hatten ja C=100 gesetzt.
Kommen wir zum zweiten Beispiel, geben Sie bitte die folgende Zeile ein:

```
1 C = 1000
```

Geben Sie nun aber bitte als Leseanfangsadresse 372 und als Endadresse 380 ein. Damit fangen wir erst nach der

Zeilennummer an zu lesen, weil diese ja gleich geblieben ist. Hier wieder die Liste mit Kurzkommentar:

Adresse	Wert	Kommentar
372	13	Zeigt an, daß eine numerische Variable vorliegt
373	5	Variablennamenlänge +4
374	0	Trennungsnull
375	227	ASCII Wert des Namens +128
376	239	Token für "="
377	26	Variablengrösse
378	232	Lowbyte des Variablenwertes
379	3	Highbyte des Variablenwertes
380	0	Trennungsnull

Wie Sie sehen, hat sich bis Adresse 376 nichts verändert, jedoch in Adresse 377 steht jetzt eine 26. Diese 26 zeigt an, daß der Wert der Variable größer als 255 ist und kleiner als 65535. Es muß jedoch immer noch eine Integerzahl sein. Das High- und Lowbyte des Variablenwertes ergeben auf die bekannte Rechenweise die Zahl 1000 und die letzte Trennungsnull ist auch bekannt.

Jetzt stellt sich also die Frage, wie der CPC Variablen mit einem Wert größer 65535 ablegt, oder wie er Zahlen mit Nachkommastellen ablegt. Dies geschieht beides auf die gleiche Art und Weise.
Geben Sie nun die folgende Zeile ein:

```
1 C = 100000
```

Geben Sie als Anfangsadresse 372 und als Endadresse 383 ein.
Die Liste sieht nun so aus:

Adresse	Wert	Kommentar
372	13	Zeigt an, daß eine mumerische Variable vorliegt
373	5	Variablennamenlänge
374	0	Trennungsnull
375	227	ASCII Wert des Variablennamens +128
376	239	Token für "="
377	31	Variablengröße
378	0	VAR 1
379	0	VAR 2
380	80	VAR 3
381	67	VAR 4
382	145	VAR 5
383	0	Trennungsnull

Bis Adresse 376 ist alles bekannt. Die 31 in Adresse 377 zeigt an, daß der Wert der Variable entweder größer 65535 ist oder keine Integerzahl ist. Die nächsten 5 Werte sind in der richtigen Form zusammengesetzt, der Wert der Variablen. Die Formel zur Errechnung des Variablenwertes ist recht komplex und nicht leicht zu durchschauen. Keine Angst, wenn Sie sie nicht verstehen.

```
Wert= (2 ^ (VAR 5 - 145)) * (65536 + (VAR 2 - 128) +
       (VAR 3 * 2) + (VAR 4 *512) + (VAR 1 - 32800))
```

KAPITEL 6 NÜTZLICHE ROUTINEN

6.1 DER JOYSTICK ALS MAUS

Wir haben schon einmal den Punkt der Ergonomie angesprochen, also der Bedienerfreundichkeit eines Programms.

Um ein Programm benutzerfreundlich zu gestalten, sind in letzter Zeit viele Konzepte und Computer entwickelt worden.

Ein wichtiger Faktor bei all diesen Methoden, ist die sogenannte "Maus". Diese ist ein kleiner Kasten mit einer Rollkugel und einer Taste. Sie können diesen Kasten auf der Rollkugel auf einer Unterlage bewegen. Analog dazu bewegt sich auf dem Bildschirm des angeschloßenen Computers ein Symbol, meistens ein kleiner Pfeil. Mit diesem Pfeil können Sie auf Felder auf dem Bildschirm zeigen, in denen bestimmte Funktionen aufgeführt sind. Drücken Sie dann den Knopf auf der Maus, führt der Computer die entsprechende Funktion aus. Sie brauchen also für solche "Manipulationen" nicht Ihre Tastatur. Nach einiger Übung können Sie mit der Maus auch viel schneller umgehen als mit der Tastatur. Es sei denn, Sie hätten das Schreiben mit allen zehn Fingern gelernt.

Eine Maus steht für den CPC noch nicht zur Verfügung. Aber man kann dieses genauso gut mit einem Joystick machen. Wir haben diesen schon einmal bei unserem Zeichenprogramm eingesetzt. Daher wissen Sie sicherlich noch, was dieser kleine Steuerknüppel für ein positives Zusatzgerät ist.

Um Ihnen die Arbeitsweise klar zu machen, haben wir eine kleine Routine geschrieben, die erst einige Menüpunkte (in unserem Beispiel einer Textverarbeitung) ausgibt. Dann können Sie mit einem angeschlossenen Joystick (Amstrad oder Atari- kompatibel) einen kleinen Punkt auf dem Bildschirm bewegen. Positionieren Sie diesen kleinen Punkt auf das

Zeichen ("O") hinter dem Menüpunkt, den Sie anwählen wollen, und drücken Sie die Feuertaste. Danach verzweigt das Programm zur entsprechenden Routine.

Nachfolgend liefern wir Ihnen erst einmal die Routine, und dann die Erklärung, sowie ein paar Veresserungsmöglichkeiten, damit Sie diese auch in Ihren Programmen einsetzen können.

Bitte beachten Sie beim Abtippen des Programmes, daß Sie die richtige Zahl von Leerzeichen in den Zeilen 30, 50, 70, 90, 110, 130, 150 und 170 eingeben !

```
10 CLS
20 LOCATE 1,1
30 PRINT"Text erstellen    O"
40 LOCATE 40,1
50 PRINT"Text ansehen      O"
60 LOCATE 1,8
70 PRINT"Text aendern      O"
80 LOCATE 40,8
90 PRINT"Text loeschen     O"
100 LOCATE 1,17
110 PRINT"Text abspeichern O"
120 LOCATE 40,17
130 PRINT"Text laden       O"
140 LOCATE 1,25
150 PRINT"Text ausdrucken  O"
160 LOCATE 40,25
170 PRINT"Programm beenden O"
180 a=JOY(0)
190 PLOT x*8,400-y*16,0
200 IF a=1 THEN y=y-1
210 IF y<1 THEN y=25
220 IF a=2 THEN y=y+1
230 IF y>25 THEN y=1
240 IF a=4 THEN x=x-1
250 IF x<1 THEN x=80
260 IF a=8 THEN x=x+1
270 IF x>80 THEN x=1
280 PLOT x*8,400-y*16,1
290 IF a=16 THEN GOTO 300 ELSE GOTO 180
300 IF x=18 AND y=1 THEN GOTO 400
310 IF x=58 AND y=1 THEN GOTO 440
320 IF x=18 AND y=8 THEN GOTO 480
330 IF x=58 AND y=8 THEN GOTO 520
340 IF x=18 AND y=17 THEN GOTO 560
```

```
350 IF x=58 AND y=17 THEN GOTO 600
360 IF x=18 AND Y=25 THEN GOTO 640
370 IF x=58  AND y=25 THEN GOTO 680
380 GOTO 180
390 END
400 CLS
410 PRINT"Text erstellen"
420 INPUT a$:IF a$<>"w" GOTO 420
430 GOTO 10
440 CLS
450 PRINT"Text ansehen"
460 INPUT a$:IF a$<>"w" GOTO 460
470 GOTO 10
480 CLS
490 PRINT"Text aendern"
500 INPUT a$:IF a$<>"w" GOTO 500
510 GOTO 10
```

Erklärung des Programms:

10-170	Ausgabe des Programmbildes mit den auszuwählenden Einzelteilen.
180	Ein Wert vom Joystickport wird in die Variable eingelesen.
190	Es wird der kleine Cursor gelöscht.
200-270	Dieses ist die Routine, um den kleinen Cursor gemäß der Bewegung des Joysticks zu verändern. Dieses geschieht, indem die Variablen für die beiden Dimensionen des Bildschirmes (x/y - Richtung) entsprechend herauf- oder herabgesetzt werden. Danach muß immer eine Abfrage eingebaut werden, ob der Wert nicht außerhalb des Bildschirmes liegt, damit später nicht einmal ein IMPROPER ARGUMENT auftritt
280	Setzen des kleinen Graphikcursors.
290	Sollte der Feuerknopf gedrückt worden sein, springt das Programm zum entsprechenden Teil, sonst wird wieder der Joystickport abgefragt
300-370	Hierhin springt das Programm, wenn der Feuerknopf

gedrückt wurde. Es wird der Reihe nach überprüft, an welcher Stelle der Cursor war, als der Feuerknopf gedrückt wurde.

380 Rücksprung zur Abfrage des Portes für den Fall, daß der Feuerknopf an einer Stelle gedrückt wurde, an der kein "O" steht

400- Ab hier können Sie dann die Routinen Ihres Programms einbauen. Wir haben hier nur ein paar kleine Beispiele eingeschrieben. Bitte beachten Sie, daß Sie die Zeilennummern in den Zeilen entsprechend anpassen.

Nachdem Sie das Programm gestartet haben, können Sie mit dem Joystick einen kleinen Cursor bewegen. In diesem Programm ist es Ihnen nur möglich, die ersten drei Punkte anzuwählen, da für die anderen keine Routinen eingebunden. Das Programm soll ja nur das Prinzip verdeutlichen.

Eine schöne Aufgabe ist es, die Routine noch an zwei Stellen zu verbessern. Erstens sollte der Cursor größer werden. Dieses können Sie erreichen, indem Sie anstelle de PLOT-Befehls mit dem LOCATE-Befehl arbeiten. Die zweite Verbesserung sollte darin bestehen, daß man den Punkt an dem man den Feuerknopf drückt, größer gestaltet. Damit ist er einfacher zu treffen. Hierfür müssen Sie in den Zeilen 300-370 einen größeren Bereich abfragen, als wir es getan haben. Das führen Sie am besten mit logischen Verknüpfungen durch.

6.2 KASSETTENOPERATIONEN

Zuerst möchten wir Ihnen einen kleinen Tip geben, wie Sie Ihre Programme oder Daten auf jeden Fall gegen Überschreiben schützen können. Dieses ist möglich, da Ihr CPC als Speichermedium normale Compactkassetten benutzt. An der hinteren Schmalseite finden Sie zwei kleine Stege.

Diese können Sie mit Hilfe eines Kugelschreibers oder ähnlichem herausbrechen. Die Folge davon ist, daß Sie die ganze Kassettenseite nicht mehr beschreiben, also auch nicht mehr überschreiben können.

Wie funktioniert das? Ganz einfach, wenn Sie eine Kassette einlegen, so greift ein kleiner Hebel hinten in diese Lasche an der Kassette. Ist diese Lasche nicht mehr vorhanden, wird der Hebel nicht zurückgedrückt und Sie können durch einen Mechanismus die RECORD-Taste nicht mehr niederdrücken und damit nicht mehr auf diese Kassette schreiben.

Um eine Seite zu schützen, müssen Sie jeweils die Lasche herausbrechen, die sich auf der linken Seite befindet, wenn Sie auf die Kassette von oben schauen.

Entschließen Sie sich später einmal, daß Sie die Kassettenseite wieder benutzen möchten, können Sie den Schreibschutz relativ einfach wieder aufheben. Sie brauchen nur einen kleinen Streifen Tesafilm über das entstandene Loch kleben. Dieser übernimmt dann die Aufgabe der Lasche, die Sie vorher herausgebrochen haben.

Achten Sie aber bitte darauf, daß der kleine Bolzen in Ihrem Kassettenrecorder auch wirklich herausspringt, da Sie sich sonst nicht auf diesen Schutz verlassen können. Bei unserem Modell (dieses war allerdings ein Prototyp) war die Feder defekt, so daß der Bolzen nicht mehr hervorsprang.

Dieses ist ganz einfach zu überprüfen, indem Sie die RECORD-Taste drücken, wenn Sie keine Kassette eingelegt haben. Läßt sich diese dann nicht drücken, ist alles in Ordnung, wenn doch, dann stimmt etwas nicht.

6.2.1 KASSETTENMOTORSTEUERUNG

Wir möchten Ihnen noch zwei Adressen mitteilen, mit denen es Ihnen möglich ist, den Kassettenmotor vom Programm aus zu steuern.

Dieses sind   Start des Kassettenmotors: &BC6E
               Stoppen des Kassettenmotors: &BC71

Möchten Sie also, daß der Motor zu laufen beginnt, geben Sie nur ein CALL &BC6E.

6.3 EINFACHER KOPIERSCHUTZ

Ein weiterer Punkt, den wir hier ansprechen möchten, ist der Schutz von Programmen gegen unerlaubtes kopieren.
Dieses ist leider zu einer weit verbreiteten Unsitte geworden, die den Autoren der Programme um einen erheblichen Teil Ihres Verdienstes bringt (von dem sie schließlich leben müssen). Außerdem ist das ein nicht unerheblicher Rechtsbruch (Es ist kein Kavaliersdelikt wie falsches parken!).

Leider lassen sich die sogenannten Raubkopierer von dieser Tatsache nicht abschrecken. Deshalb werden sogenannte Kopierschutzmechanismen entwickelt, die verhindern sollen, daß Programme unerlaubt vervielfältigt werden.

Ihr CPC hat einen solchen Kopierschutz schon eingebaut ! Dieser verbirgt sich ganz unscheinbar unter dem Befehl SAVE.

Wie ist dieser Kopierschutz zu handhaben? Wenn Sie ein Programm abspeichern, das geschützt werden soll, brauchen Sie an den Programmnamen nur ein Komma und ein "P" (für protected = geschützt) anzuhängen.
Ein Programm, was auf diese Weise abgespeichert wurde, ist nur mit "RUN" zu laden. Es startet dann automatisch. Sie können dann das Programm zwar unterbrechen, aber damit ist es für Sie "verschwunden". Es ist nicht möglich, dieses Programm zu listen, noch es abzuspeichern oder ähnliches.

6.4 ABSPEICHERN EINES SPEICHERBEREICHES

Eine weitere nützliche Hilfe steht ebenfalls etwas abseits im Handbuch. Dieses ist ein Befehl, um bestimmte Speicherbereiche abzuspeichern. Sie können diesen Befehl zum Beispiel in einem Assembler benutzen, um bestimmte Bereiche auf Kassette abzulegen (wir haben dieses bei unserem Monitor getan).

Eine weitere phantastische Möglichkeit dieses Befehles ist es, den ganzen Bildschirm abzuspeichern (oder Teile von diesem). Wir haben uns diese Eigenschaft bei dem Grafikeditor zunutze gemacht. Sie können diese Bereiche natürlich auch wieder laden und somit relativ schnell phantastische Bilder auf Ihrem Monitor erscheinen lassen, ohne mit den langsameren Graphikbefehlen des CPC arbeiten zu müssen. Für den Bildschirm schauen Sie sich bitten den Speicherbereich an, in dem dieser liegt. Die beiden Adressen haben wir Ihnen schon in einem der vorigen Teilkapitel mitgeteilt.

6.5 SINNVOLLE TASTENBELEGUNG

Ihr CPC hat die Möglichkeit, mit zwei einfachen Befehlen die Belegung der Tasten zu verändern. Dieses sind KEY und KEY DEF. Wir liefern Ihnen nachfolgend ein Programm, das die Belegung der Tastatur verändert. Im weiteren erklärt sich das Programm von selbst.

```
10 MODE 2
20 LOCATE 22,5:PRINT"K  E  Y      M  A  N  A  G  E  R"
30 LOCATE 22,6:PRINT"================================="
40 PRINT:PRINT:PRINT:PRINT:PRINT
50 PRINT"Folgendes Programm belegt den Zehnerblock Ihrer Tas
tatur mit einigen Hilfsbefehlen,sowie einige Tasten mit deut
schen Sonder
zeichen"
60 PRINT:PRINT"Bitte eine Taste druecken."
70 CALL &BB18
80 CLS
90 PRINT:PRINT"Der Zehnerblock wird wie folgt mit Befehlen b
elegt :"
100 PRINT:PRINT:PRINT
110 KEY 137,d$
120 PRINT"Taste 0  :   LIST"
130 PRINT"Taste 1  :   RUN"
140 PRINT"Taste 2  :   LOAD"
150 PRINT"Taste 3  :   MODE"
160 PRINT"Taste 4  :   SAVE"
170 PRINT"Taste 5  :   EDIT"
180 PRINT:PRINT:PRINT
190 PRINT"Ausserdem haben Sie die Moeglichkeit fuenf weitere
 Tasten mit einer Zeichenfolge von jeweils 15 Zeichen zu bel
egen."
200 INPUT"Wenn Sie dieses moechten,geben Sie bitte ein 'd'";
d$
210 IF d$="d" THEN GOTO 280
220 CLS
230 PRINT"Ausserdem stellt Ihnen dieses Programm die deutsch
en Sonderzeichen zur Verfuegung, und es stellt die y und z T
aste um"
240 PRINT:PRINT:PRINT:PRINT
250 PRINT"Bitte eine Taste druecken"
260 CALL &BB18
270 GOTO 350
280 CLS
```

```
290 INPUT"Belegung der Taste 6";a$
300 INPUT"Belegung der Taste 7";b$
310 INPUT"Belegung der Taste 8";c$
320 INPUT"Belegung der Taste 9";d$
330 INPUT"Belegung der Taste '.'";e$
340 GOTO 220
350 KEY 128,"list"+CHR$(13)
360 KEY 129,"run"+CHR$(13)
370 KEY 130,"load"+CHR$(13)
380 KEY 131,"mode "
390 KEY 132,"save"+CHR$(13)
400 KEY 133,"edit "
410 KEY 134,a$
420 KEY 135,b$
430 KEY 136,c$
440 KEY 137,d$
450 KEY 138,e$
460 SYMBOL AFTER 32
470 SYMBOL 59,0,20,0,31,34,66,66,127
480 SYMBOL 58,0,36,0,60,66,66,66,60
490 SYMBOL 60,0,0,36,0,66,66,66,60
500 SYMBOL 62,129,60,66,66,126,66,66,66
510 SYMBOL 64,129,60,66,66,66,66,66,66
520 SYMBOL 36,129,0,66,66,66,66,66,60
530 SYMBOL 39,60,68,68,120,68,124,64,64
540 SYMBOL 122,0,36,36,36,28,4,56
550 SYMBOL 90,0,65,34,20,8,8,8,8
560 SYMBOL 121,0,0,15,1,1,2,4,15
570 SYMBOL 89,64,2,4,8,8,16,16,31
```

6.6 SPRUNGADRESSEN

Auf den folgenden Seiten werden Sie einige nützliche Adressen finden, die Sie in Ihre Programme mit einbinden können. Um diese Systemroutinen nutzen zu können, müssen Sie kein Maschinensspracheprogrammierer sein. Es reicht vollkommen aus, wenn Sie das Kapitel 4 mit der Einführung in die Maschinensprache gelesen haben. Dem interessierten Leser werden kurz die Funktion und die benötigten Eingabe/Ausgabe Register beschrieben. Selbstverständlich werden Sie auch auf etwaige Besonderheiten der Routinen (soweit sie bekannt sind) hingewiesen. Nachfolgend sei noch einmal der Basic Lader aufgeführt, mit dem sich vom Basic aus alle Funktionen aufrufen lassen.

```
10 REM Basic Lader
20 MEMORY &2FFF : d = 0
30 INPUT "Wert ";w
40 IF w = 0 THEN GOTO 80
50 POKE &3000+d,w
60 d = d + 1
70 GOTO 30
80 CALL &3000
```

Der oben angeführte Basic Lader ist gegenüber dem aus dem Kapitel 4 leicht abgeändert. Es handelt sich um eine "Endlosschleife" mit einem Abbruchkriterium. In Zeile 40 wird abgeprüft, ob der eingegebene Wert gleich 0 ist. Wenn das der Fall ist, dann wird das mittels Lader erzeugte Unterprogramm sofort aufgerufen. Sollten Sie einmal in die Verlegenheit kommen, daß Sie eine 0 als Datenwert eingeben müssen, dann ändern Sie den Wert in Zeile 40, worauf hin der Abbruch der Werteeingabe veranlaßt wird.
Wenn Sie einige der Funktionen in ein Programm einbinden wollen, so empfiehlt es sich, die Werte des Basic Laders in DATA Zeilen abzulegen, mit denen dann das entsprechende Unterprogramm aufgebaut wird. Variable Werte lassen sich mit einem Poke Befehl an die richtige Speicherstelle bringen.

6.6.1 BEREICH EINFÄRBEN

Einsprungadresse : BC44 (hex)
48196 (dec)

Mit dieser Routine lassen sich Bildschirmbereiche einfärben. Die Größe ist abhängig von dem gewählten Betriebsmodus und den frei wählbaren Grenzpunkten.

Übergaberegister: a codierter Farbwert (siehe auch Kapitel 2.10 Bildschirmspeicher)
h linke Spalte des zu füllenden Bereiches
d rechte Spalte des Bereiches
l oberste Reihe des Bereiches
e unterste Reihe des Bereiches

Achten Sie darauf, daß Sie mit den Bereichsgrenzen nicht über den Bildschirm hinauskommen, da andernfalls Ihr CPC "abstürzen" könnte. Die linke obere Bildschirmecke hat den physikalischen Wert 0,0.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 3E, (Wert für Zeichenfarbe/Bitmuster), 21, (oben), (links), 11, (unten), (rechts), CD, 44, BC, C9, 0
Die Werte für oben, links ect. sind frei wählbar und entsprechen den Zeilen oder Spaltenzahlen.

Assemblerformat :
```
ld   a,N
ld   hl,NN
ld   de,NN
call 44 BC
ret
```

6.6.2 BILDSCHIRM BANK WECHSELN

Einsprungadresse : BC06 (hex)
48134 (dec)

Mit dieser Routine kann der Bildschirmspeicher auf eine der vier vorhandenen Banks (0-3) gelegt werden. Sinnvoll ist hier aber nur die Bank 1 oder die Standard Bank 3, da es an den anderen beiden Stellen zu Konflikten mit dem Betriebssystem des CPC kommen kann.

Übergaberegister: a Im Akkumulator wird das signifikante Byte (Higher-Byte) der Startadresse des neuen Bildschirmbereiches übergeben.
Beachten Sie bitte, daß Sie nur die Werte 00, 40, 80 und C0 verwenden, da es andernfalls zu nicht gewünschten Auswirkungen kommen kann.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 3E, (neuer Startwert), CD, 06, BC, C9, 0
Beachten Sie bei dem neuen Startwert die Hinweise, die Sie in Übergaberegister finden.

Assemblerformat : ld a,N
call 06 BC
ret

6.6.3 WARTEN AUF TASTE

Einsprungadresse : BB06 (hex)
47878 (dec)

Der Ansprung dieser Routine bewirkt eine Unterbrechung des Progammlaufs. Es wird gewartet, bis eine beliebige Taste gedrückt ist.

Übergaberegister----

Übernahmeregister:a Aus dem Akkumulator kann der Code der gedrückten Taste ausgelesen werden.

Hexadezimalwerte für den Basic Lader ---
Diese Funktion kann an beliebiger Stelle im Programm mit CALL &BB06 aufgerufen werden.

Assemblerformat : call 06 BB

6.6.4 SCROLLING

Einsprungadresse : BC4D (hex)
48205 (dec)

Durch Aufruf dieser Routine ist es möglich, den kompletten Bildschirm um 1 Zeile nach oben oder unten zu verschieben. Der Farbwert (PAPER) der neu eingefügten Zeile kann innerhalb der Möglichkeiten des gewählten MODE frei bestimmt werden.

Übergaberegister: b Durch Register b wird die Verschieberichtung festgelegt. Hat der Inhalt den Wert 0, so wird abwärts gescrollt. Jeder andere Wert bewirkt eine Aufwärtsverschiebung.

a Im Akkumulator wird der codierte Farbwert für die neu erscheinende Zeile angegeben.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : (aufwärts) 6, 1, 3E, (Farbcode der neu einzufügenden Zeile), CD, 4D, BC, C9, 0
Hexadezimalwerte für den Basic Lader : (abwärts) 6, 0, 3E, (Farbcode der neu einzufügenden Zeile), CD, 4D, BC, C9, 0

Assemblerformat :
```
ld   b,N
ld   a,N
call 4D BC
ret
```

6.6.5 SCROLLING VON BILDSCHIRMTEILEN

Einsprungadresse : BC50 (hex)
48208 (dez)

Über den Aufruf dieser Routine sind Sie in der Lage, Teilbereiche (Fenster) zu verschieben. Verschieberichtung und Farbcode der neuen Zeile entsprechen den Registern des Scrollings für den kompletten Bildschirm.

Übergaberegister : a codierter Farbwert
b Verschieberichtung
h linke Spalte des zu verschiebenden Fensters
d rechte Spalte des Fensters
l oberste Reihe des Fensters
e unterste Reihe des Fensters

Achten Sie bitte darauf, daß die Bildschirmgrenzen nicht überschritten werden, da es andernfalls zu Komplikationen kommen kann. Die linke, obere Bildschirmecke hat den physikalischen Wert 0,0.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 6, (Richtung), 3E, (Farbcode), 21, (oben), (links), 11, (unten), (rechts), CD, 50, BC, C9, 0
Die gewählten Bereichsgrenzen sind von keinem vorher definierten Fenster abhängig.

Assemblerformat :

```
ld   b,N
ld   a,N
ld   hl,NN
ld   de,NN
call 50 BC
ret
```

6.6.6 MODE ANWÄHLEN

Einsprungadresse BCOE (hex)
48142 (dez)

Über diese Routine ist es Ihnen möglich, in Maschinenprogramm den Bildschirmmode zu wechseln.

Übergaberegister : a Der Akkumulator enthält den Wert des neu zu wählenden Mode
Der Wert des Akkumulators darf nur zwischen 0 und 2 liegen.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 3E, (neuer Mode), CD, E, BC, C9, 0

```
Assemblerformat : ld   a,N
                  call OE BC
                  ret
```

6.6.7 INVERTIEREN EINES ZEICHENS

Einsprungadresse BC4A (hex)
48202 (dez)

Mit dieser Routine kann ein Zeichen, welches durch seine physikalischen Koordinaten festgelegt wird, (0,0-linke obere Ecke) über 2 Bitmustermasken invertiert werden. Die Invertierung wird über ein zweifaches logisches EXCLUSIVE OR erreicht.

Übergaberegister : b Invertierungsmaske 1 (Farbe 1)
c Invertierungsmaske 2 (Farbe 2)
h Spalte
l Reihe

Es ist zu beachten, daß die Position vom jeweils gewählten Mode abhängig ist.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 6, (Maske 1), E, (Maske 2), 21, (Spalte), (Zeile), CD, 4A, BC, C9, 0

Beispiel: Mode 1, Zeichenfarbe gelb, Hintergrundfarbe blau. Das auf dem Bildschirm an der zu invertierenden Position ausgegebene Zeichen ist eine "0". Betrachten wir die Invertierung der beiden obersten Bitmusterzeilen. Als Invertierungsmaske 1 wählen wir "1000 0000", Maske 2 hat dieses Aussehen : "0000 1111".

Ausgangssituation 0111 0000	1100 0000	
Nach Invertierung 1.....1111 0000	0100 0000	
Nach Invertierung 2.....1111 1111	0100 1111	
Farbe der Pixle.........RRRR RRRR	HHRR HHHH	R = Rot
		H = Hellblau

```
Assemblerformat : ld   b,N
                  ld   c,N
                  ld   hl,NN
                  call 4A BC
                  ret
```

6.6.8 HORIZONTALE BILDSCHIRMVERSCHIEBUNG

Einsprungadresse BC05 (hex)
48233 (dez)

Diese Routine ermöglicht es Ihnen, den Bildschirm horizontal zu verschieben (rechts links Scrolling). Dabei wird die links herausfallende Spalte an der rechten Bildschirmseite angefügt (wrap arround).

Übergaberegister : h Higher Byte des Verschiebewertes
l Lower Byte des Verschiebewertes
Es ist darauf zu achten, daß der Verschiebewert grundsätzlich durch 2 teilbar sein muß.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 21, (Lower Byte des Verschiebewertes), (Higher Byte des Verschiebewertes), CD, 5, BC, C9, 0

```
Assemblerformat : ld   hl,NN
                  call 05 BC
                  ret
```

6.6.9 CURSOR POSITIONIEREN

Einsprungadresse BB75 (hex)
47989 (dez)

Diese Routine entspricht in ihrer Position dem Basicbefehl "LOCATE"

Übergaberegister : h neue Cursorspalte
l Cursorzeile

Eine Überprüfung, ob die gesetzten Werte innerhalb des Bildschirmes liegen, findet nicht statt.

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 21, (Zeile), (Spalte), CD, 75, BB, C9, 0

Assemblerformat :
```
ld   hl,NN
call 75 BB
ret
```

6.6.10 SPALTENPOSITIONIERUNG DES CURSORS

Einsprungadresse BB6F (hex)
47983 (dez)

Über diese Routine ist die Positionierung des Cursors innerhalb der Zeile, in der er sich befindet, möglich.

Übergaberegister : a neuer Spaltenwert
Eine Plausibilitätskontrolle auf den gesetzten Spaltenwert findet nicht statt

Übernahmeregister----

Hexadezimalwerte für den Basic Lader : 3E, (Spalte), CD, 6F, BB, C9, 0

Assemblerformat :
```
ld   a,N
call 6F BB
ret
```

6.6.11 ZEILENPOSITIONIERUNG DES CURSORS

Einsprungadresse : BB72 (hex)
47986 (dec)

Über diese Routine ist die Positionierung des Cursors innerhalb der Spalte, in der er sich befindet, möglich.

Übergaberegister : a neuer Zeilenwert
Eine Überprüfung, ob die neu gesetzte Spalte vorhanden ist, findet nicht statt

Übernahmeregister ---

Hexadezimalwerte für den Basic Lader : 3E, (neue Zeile), CD, 72, BB, C9, 0

Assemblerformat :

```
ld   a,N
call 72,BB
ret
```

6.6.12 JOYSTICK ABFRAGEN

Einsprungadresse : BB24 (hex)
47908 (dec)

Hier handelt es sich um die Maschinenroutine, mit der die geschlossenen Schalter der beiden anschließbaren Joysticks abgefragt werden.

Übergaberegister ---

Übernahmeregister : a Nach der Abfrage befindet sich der Code der geschlossenen Kontakte im Akkumulator.
h Gleiche Belegung wie der Akkumulator
l Der Status des 2. Joysticks wird im Register l zwischengespeichert.

Ein gesetztes Bit entspricht dabei einem geschlossenen Kontakt. Folgende Schalterstellungen sind den einzelnen Bits zugeordnet :

Bit 0	Oben
Bit 1	Unten
Bit 2	Links
Bit 3	Rechts
Bit 4	Feuer 2 (Standardfeuerknopf)
Bit 5	Feuer 1
Bit 6	unbenutzt
Bit 7	unbenutzt

Hexadezimalwerte des Basic Laders : CD, 24, BB, 32, (lower Byte Feuer 2), (higher Byte Feuer 2), 7D, 32, (lower Byte Feuer 1), (higher Byte Feuer 1), C9, 0
Nachdem diese Routine aufgerufen wurde, können Sie mit zwei PEEK Befehlen den Status beider Joysticks abfragen.

```
Assemblerformat :   call 24,BB
                    ld   (NN),a
                    ld   a,l
                    ld   (NN),a
                    ret
```

6.6.13 INVERS EIN- UND AUSSCHALTEN

Einsprungadresse : BB9C (hex)
48028 (dec)

Über diese Routine werden die aktuellen Farben des PAPERs und des PENs gegeneinander ausgetauscht. Mit dem ersten Aufruf wird "Invers" eingeschaltet, mit dem nächsten Aufruf wieder abgeschaltet. In Basic ist diese Funktion mit dem Ausdruck : Print Chr$(24) zu erreichen.

Übergaberegister ---

Übernahmeregister ---

Hexadezimalwerte des Basic Laders ---
Ein Aufruf ist über CALL &BB9C möglich.

Assemblerformat : call 9C,BB

6.7 RANDOMIZE - DEM ZUFALL AUF DER SPUR

Ihr CPC verfügt über zwei Befehle, mit denen Sie den sogenannten Zufall steuern können. Das ist einmal die RND Funktion. Mit ihr wird der nächste Wert aus einer Zufallszahlenreihe ausgegeben. Probieren Sie das folgende Beispiel, dann werden Sie sehen, was es mit dem Zufall auf sich hat. Zuvor müssen Sie Ihren Rechner aber noch mit CTRL + Shift + ESC zurücksetzen, damit wir die gleichen Voraussetzungen haben.

```
10 REM Zufall?
20 FOR t = 1 to 5
30 PRINT RND
40 NEXT t
```

Vergleichen Sie doch Ihre Zahlen mit denen, die Sie hier nachfolgend sehen. Sie werden feststellen, daß es keinen Unterschied gibt.

```
0.271940658
0.528612386
0.021330127
0.175138616
0.657773343
```

Das liegt darin begründet, daß die Zufallszahlenreihe mit jeder Initialisierung des Computers wieder zurückgesetzt wird. Stellen Sie sich einmal vor, Sie würden ein Spiel schreiben oder auch kaufen, bei dem Sie schon nach kurzer Zeit die Aktionen Ihrer Gegner voraussehen können. Dieses Spiel würde sicherlich sehr schnell langweilig werden.
So gibt es einen weiteren Befehl, der sich RANDOMIZE xx nennt. Für xx müssen Sie einen Zahlenwert eintragen. Die Zufallszahlenfolge beginnt dann mit einem Wert, der abhängig von dem bei RANDOMIZE vorgegebenen Startwert ist. Dazu ein weiterer Versuch. Versetzen Sie den CPC in gewohnter Weise

in seinen Einschaltzustand. Anschließend geben Sie das Programm "Zufall?" ein. Wir wollen aber auch die RANDOMIZE Funktion berücksichtigen. Fügen Sie deshalb noch eine weitere Zeile in Ihr Programm ein :

```
15 RANDOMIZE 33
```

Starten Sie mit "RUN" und notieren Sie sich die so ermittelten Zufallszahlen. Einen gewissen Teilerfolg haben wir sicherlich erreicht, da die neue Zufallszahlenreihe keine Ähnlichkeit mit der hier abgedruckten aufweist. Nur, ist damit das Problem behoben, daß nach einem RESET immer die gleichen Zahlen aufgerufen werden? Nein!
Wenn wir eine echte Zufallszahl erzeugen wollen, so müssen wir einen Faktor einschalten, den wir nicht absehen können. Hier bietet sich der Zeitgeber Ihres Computers an. Außer den vier Zeitgebern, mit denen Sie Interrupts aktivieren, steht Ihnen noch eine Echtzeituhr zur Verfügung. In der Echtzeituhr wird die Zeit fortgeschrieben, die seit Einschalten des Gerätes vergangen ist. Besser noch, die Zeit wird in vergangenen 3/100 Sekunden gemessen. Hier greifen wir an und setzen mit der RANDOMIZE Funktion den RND Startwert auf eine Zahl, die durch die vergangene Zeit bestimmt wird. Probieren Sie das nachfolgende Musterprogramm mehrmals aus. Sie werden feststellen, daß es praktisch keine Wiederholungen gibt. Sollten Sie doch einmal eine identische Reihe bekommen, dann kann man wirklich von Zufall sprechen.
Hier die Grundfunktion für den echten Zufall :

```
10 REM Zufall
20 RANDOMIZE TIME
30 FOR t = 1 to 5
40 PRINT RND
50 NEXT t
```

Kernstück ist die Zeile 20, in der die RANDOMIZE Funktion mit der Zeit gesetzt wird. Die Variable TIME ist bei Ihrem

Computer für diese Funktion reserviert.
RND wird immer eine Zahl im Bereich von größer 0 und kleiner 1 liefern. Wenn Sie ganze Zahlen in vorgegebenen Bereichen damit erzeugen wollen, so muß dieser Grundwert modifiziert werden. Nehmen wir als Beispiel an, daß Sie Zufallszahlen im Bereich zwischen 2 und 12 erzeugen wollen. Der Wert 12 ist die obere Bereichsgrenze, die 2 ist der untere Grenzwert. Der obere Grenzwert abzüglich dem unteren zuzüglich 1 ergibt die Anzahl der möglichen Werte. Bei uns ist das : 12 - 2 + 1 = 11. Die von RND gelieferte Zufallszahl wird mit dem so ermittelten Mengenfaktor multipliziert. In Anbetracht der Tatsache, daß RND niemals den Wert 1 liefert, werden unsere modifizierten Zufallszahlen jetzt in der Größenordnung von 0 bis 10 zu suchen sein. Der letzte Schritt bewirkt, daß der Bereich auf den kleinsten Anfangswert geschoben wird. Dazu wird die untere Bereichsgrenze zu der ermittelten Zahl addiert. Hier die Formel :

```
x = INT (RND * (Obergrenze - Untergrenze + 1) + Untergrenze
```

Mit der Funktion INT wird abschließend der ganzzahlige Anteil der so gewonnenen Zufallszahl extrahiert.

6.8 DER CPC ALS RECHENMASCHINE

In den Anfängen des Computerbaus war das einzige Ziel der Konstrukteure, einen Rechner zu bauen, der schwierige Berechnungen schneller als der Mensch anfertigen konnte. Diese Eigenschaft hat auch heute noch jeder Computer, jedoch tritt sie, bei den Homecomputern, meist in den Hintergrund. Wir wollen den Lesern unter Ihnen, die den CPC hauptsächlich für Rechnungen benutzen, einige Anregungen geben, wie Sie diese einfacher durchführen können.

Wir werden im folgenden zuerst die Bedienung des CPC als "Taschenrechner" durch ein kleines Programm vereinfachen. Sie haben sicher schon bemerkt, daß es recht lästig ist, bei einfachen Rechnungen, wie eine Addition, immer mit der SHIFT-Taste arbeiten zu müssen. Aus diesem Grund ändert unser Programm auch die Anordnung der Rechenzeichen auf der Tastatur, damit weder die SHIFT noch mehrere Tasten auf einmal gedrückt werden müssen. Dies geschieht mittels der Funktion KEY DEF.

Sollte es Ihnen nichts ausmachen mit der oberen Zahlenreihe zu arbeiten, anstatt mit dem Zahlenblock, so können Sie die Unterroutine ab 200 getrost abschreiben, da diese nämlich auf dem Zahlenblock, die auf einem normalen Taschenrechner festverdrahteten Funktionen (SIN(X);COS(X)), generiert.

Die Benutzung des Programmes ist recht einfach. Nach dem Starten meldet sich Ihr CPC nach einer kurzen Zeit mit der Meldung "UMSTELLUNGEN VORGENOMMEN". Nun sind alle Tasten mit mathematischen Zeichen zur leichteren Bedienung umgestellt worden. Zusätzlich ist die Taste für das "P" mit dem "?" belegt worden, um Ihnen das Eingeben des Befehles PRINT, beziehungsweise das Drücken der Divisionstaste in Verbindung mit SHIFT abzunehmen.
Die nachfolgende Tabelle zeigt im einzelnen, welche Umstellungen vorgenommen worden sind.

Mathematische Zeichen	Taste	Tastenwert
*	:	29
+	;	28
(		17
)		19
&		26
SIN(	0	128
COS(	1	129
TAN(	2	130
ATN(	3	131
ABS(	4	132
INT(	5	133
HEX$(	6	134
BIN$(	7	135
STR$(	8	136
?	P	27

Eine gute Bedienungshilfe wäre es auch, wenn Sie sich bei mehrfacher Anwendung kleine Aufkleber mit den oben genannten Zeichen anfertigen und diese auf die entsprechenden Tasten kleben würden, damit Sie nicht dauernd nach der Taste suchen müssen.

```
10 REM Taschenrechner
20 MODE 2
30 KEY DEF 29,1,42
40 KEY DEF 28,1,43
50 KEY DEF 17,1,40
60 KEY DEF 19,1,41
70 KEY DEF 26,1,38
80 KEY DEF 27,1,63
90 GOSUB 200
100 PRINT" Umstellungen vorgenommen"
110 END
```

```
200 KEY 128,"sin("
210 KEY 129,"cos("
220 KEY 130,"tan("
230 KEY 131,"atn("
240 KEY 132,"abs("
250 KEY 133,"int("
260 KEY 134,"hex$("
270 KEY 135,"bin$("
280 KEY 136,"str$("
290 RETURN
```

Erklärungen zum Listing:

10	Titel
20	Einstellen des 80-Zeichenmodus
30-80	Umstellen der Tasten auf die mathematischen Zeichen mittels des Befehles KEY DEF
90	Sprung zur Unterroutine ab 200, wenn die Unterroutine weggelassen wird, entfällt diese Zeile
100	Ausgabe der Programmendemeldung
110	Programmende
200-280	Umdefinieren des Zahlenblockes zu Funktionstasten mittels des Befehles KEY
290	Rücksprung zum Hauptteil

Sollte Ihnen die Auswahl der Funktionen nicht zusagen, so können diese in den Zeilen 200-290 geändert werden. Ebenso die Tastenbelegung der mathematischen Zeichen in den Zeilen 30-80 kann geändert werden, dafür müssen Sie die Tastenwerte nach Ihren Vorstellungen ändern. Die Tastenwerte finden Sie im Anhang Ihres CPC Handbuches.

6.8.1 RECHENGENAUIGKEIT

Unter der Rechengenauigkeit eines Computers versteht man die Anzahl der Nachkommastellen, die mit dem Standartwert der Berechnung übereinstimmen. So hat zum Beispiel ein Rechner, der PI als 3.1416 ausgibt, keine hohe Rechengenauigkeit, da die festgesetzten Werte, wie zum Beispiel PI oder die Wurzel einer Zahl nicht genau genug definiert sind.
Ihr CPC kann "nur" Zahlen bis zu neuneinhalb Stellen hinter beziehungsweise vor dem Komma korrekt speichern, das heißt, das bei größeren Zahlen die Ungenauigkeit größer als 1 ist. Dieses wollen wir an einem Beispiel demonstrieren.:

Geben Sie ein: PRINT 5E9+1-5E9

Sie erwarten sicher, das als Ergebnis eine eins erscheint, dem ist aber nicht so. Die Ungenauigkeit der Berechnung beläuft sich, in unserem Fall, auf eins, da als Ergebnis eine zwei auftaucht.
Die größte Integerzahl die korrekt gespeichert werden kann, ist 2^32-1 , nicht exponential ausgedrückt:
 4294967295

Diese Tatsache hängt mit der Abspeicherung von Variablen zu tun, die wir in Kapitel 5 besprochen haben.

Um die Rechengenauigkeit zu erhöhen, besteht die Möglichkeit, durch Formeln die benötigten Werte zu ermitteln. Jedoch dürfen in diesen Formeln keinerlei feste Werte, wie Wurzeln oder PI vorkommen, da diese der Genauigkeit nur schaden.
Eine solche Formel haben wir zu einem Programme umgebaut, so daß Sie dieses nur noch abtippen müssen. Sie soll Ihnen als Beispiel dienen, wie Sie Ihre eigenen Formeln in Programmen einsetzen können, ohne den Befehl DEF FN zu benutzen. Denn in einigen Fällen kann man diesen nicht anwenden.

Die Formel berechnet EXP(X) auf folgende Weise:

EXP(X)=1+X/1!+X^2/2!+X^3/3!....

Es besteht mit unserem Programm die Möglichkeit, die obige Reihe bis zur Fakultät 31 auszurechnen, danach ist die Grenze von 1E34 erreicht.
Da der CPC die Funktion der Fakultät nicht besitzt, haben wir diese in der Unterroutine ab 100 simuliert.
Die Bedienung des Programmes ist denkbar einfach, man braucht nur den gewünschten X-Wert eingeben und erhält nach einer kurzen Rechenzeit den EXP(X) ausgeben.

```
10 REM EXP(X)
20 INPUT"x-WERT";x
30 FOR n=1 TO 31
40 x1=x^n
50 GOSUB 100
60 erg1=x1/fa
70 erg2=erg2+erg1
80 NEXT
90 PRINT erg2+1
95 END
100 fa=1
110 FOR m=1 TO n
120 fa=fa*m
130 NEXT m
140 RETURN
```

Erklärungen zum Listing

10	Titel
20	Eingabe des X-Werte
30	Eröffnen der Schleife, die die Anzahl der Fakultäten bestimmt
40	Bildung der X-Potenzen
50	Sprung zur Unterroutine ab 100
60	Verarbeiten der Fakultäten
70	Addieren des Ergebnisses

80   Schließen der Schleife
90   Ausgabe des Ergebnisses
95   Beenden des Programmes
100   Zurücksetzen der Fakultät
110   Eröffnen der Schleife für die Fakultät
120   Formel zur Berechnung der Fakultät
130   Schließen der Schleife
140   Rücksprung in den Hauptteil

Eine solche Formel, wie die von EXP(X), kann man sicher nur ungenau mit einem DEF FN Befehl ausrechnen, da man nicht alle 31 Teile in den DEF FN Befehl einbringen kann.
Ein weiteres Beispiel für den Einsatz von Formeln, ist die Berechnung von Nullstellen. Nullstellen sind die Punkte, an denen der Graph einer Funktion die X-Achse schneidet. Es sind sehr wichtige Punkte zur Analyse eines Funktionsgraphen, die man nie genau aus einer Zeichnung entnehmen kann.
Wir haben als Beispiel einer solchen Nullstellenberechnung eine quadratische Gleichung genommen. Diese hat die Form:

```
F(X)=X^2+P*X+Q
```

Die Formel zur Berechnung der Nullstellen lautet so:

```
X1= -P/2+SQR(P^2/4*Q); X2= -P/2-SQR(P^2/4*Q)
```

Zur Bedienung ist zu sagen, daß Sie zuerst den Wert für P und dann den Wert für Q eingeben müssen. Danach gibt das Programm den Wert für die Nullstellen (X1,X2) aus.

```
10 INPUT p
20 INPUT q
30 x1=-p/2+SQR(p*p/4-q)
40 x1=-p/2-SQR(p*p/4-q)
50 IF x1*x2<>q THEN 80
60 PRINT x1,x2
70 END
80 PRINT"Keine Nullstellen"
90 END
```

Erklärungen zum Listing

10	Eingabe P
20	Eingabe Q
30	errechnen der ersten Nullstelle
40	errechnen der zweiten Nullstelle
50	Nachprüfen der Nullstellen durch den Satz von Vieta
60	Ausgabe der Nullstellen
70	Programmende
80	Ausgabe, da keine Nullstelle vorhanden

Wird das Programm durch eine Fehlermeldung unterbrochen, so ist keine Nullstelle vorhanden.

6.8.2 RECHENGESCHWINDIGKEIT

Ein sehr wichtiges Merkmal zur Beurteilung eines Computers ist seine Rechenzeit. Es hilft dem Anwender die Eigenschaften des Rechners richtig einzuschätzen.
Zumeist wird die Rechengeschwindigkeit mit einem Testprogramm ermittelt, das auf mehreren Rechnern gelaufen ist. Auf diese Art hat man Vergleichswerte zwischen den Rechnern, und kann entscheiden, welcher für die gewünschte Anwendung geeignet ist.

Es gibt mehrere Aufgaben, die man einem Rechner stellen kann, um die Rechengeschwindigkeit zu erfahren. Alle Standardaufgaben werden unter dem Begriff BENCHMARK-Test geführt. Einen solchen BENCHMARK-Test haben auch wir in diesem Kapitel verwandt. Er nennt sich:"Sieb des Eratosthenes"; und ist ein sehr häufig verwandte Methode zur Bestimmung von Geschwindigkeitsunterschieden zwischen Compilern.

Zur Rechengeschwindigkeit gehört auch die Abarbeitung von Abfragen, Schleifen und die Verwaltung von Datenfeldern. Alle diese Aufgaben bearbeitet das "Sieb des Eratosthenes". Zum Vergleich haben wir zwei Rechner diese Aufgabe durchrechnen lassen. Die Ergebnisse sahen folgendermaßen aus:

Apple-II-Europlus mit MBASIC 5.2 benötigte 391,8 sec.
VC-20 mit 64K Karte benötigte 394.8 sec.

Damit Sie die Zeit Ihres CPC selbst testen können, haben wir Ihnen das Listing aufgeführt. Lassen Sie das Programm laufen, stoppen Sie die Zeit und vergleichen Sie selbst, ob Ihr CPC den Vergleich aufnehmen kann.

```
10 REM Eratosthenes
20 DEFINT a-z
30 DIM fl(1000):l=1000:PRINT"Start"
40 FOR j=1 TO 10
50 co=0
60 FOR i=1 TO l:fl(i)=1:NEXT i
70 FOR i=1 TO l
80 IF fl(i)<>1 THEN 160
90 prim=i+i+3
100 k=i+prim
110 IF k>l THEN 150
120 fl(k)=0
130 k=k+prim
140 GOTO 110
150 co=co+1
160 NEXT i
170 NEXT j
180 PRINT co;"Primzahlen"
190 END
```

Eine ausführliche Erklärung zum Listing möchten wir nicht geben, da diese recht komplex und dadurch schwierig ausfallen würde.

Sie werden nach diesem Test sehen, das Ihr CPC auch auf dem Gebiet Geschwindigkeit, es mit anderen Rechnern aufnehmen kann.

Noch ein kleiner Tip am Rande: Ihr CPC besitzt eine Vielzahl von mathematischen Sonderzeichen. Sie können diese jedoch nur über PRINT CHR$(X) erreichen, das heißt, sie sind von der Tastatur nicht direkt erreichbar. Es dürfte allerdings nicht zu schwer sein, diese in einem Programm so zu verarbeiten, daß die Sonderzeichen ihre eigentliche Funktion erfüllen. Ein Beispiel dafür wäre die Zahl PI.

6.9 BILDSCHIRMBEWEGUNGEN

In manchen Basicdialekten gibt es einen Befehl SCROLL, der es erlaubt, den Bildschirm um eine Zeile nach oben oder unten zu rollen. Wir haben eine Routine des CPC Rom ausgenutzt, um diesen Befehl zu simulieren.
Der Befehl SCROLL kann zum Beispiel in Spielen eingesetzt werden oder beim Auflisten einer Tabelle.
Die normale Art den Bildschirm zu bewegen ist es, in der rechten unteren Ecke ein Zeichen einzufügen, damit der Bildschirm sich nach oben rollt. Jedoch fehlt die Möglichkeit, den Bildschirm auch nach unten zu rollen. Meist benötigt man hierzu eine recht lange Maschinensprachroutine, um dies zu erzeugen. Beim CPC besteht aber die Möglichkeit, auf Routinen des Roms zurückzugreifen.

Die Routine für die Bewegung des Bildschirms steht ab &BC4D im Rom (siehe auch Kapitel 6.6 Sprungadressen). Sie bewegt den Bildschirm um acht Pixel (ein Zeichen). Die Richtung der Bewegung hängt von dem Wert des B-Registers ab. Ist das B-Register gleich null, so ist die Bewegung nach unten gerichtet. Bei allen anderen Werten (ungleich null), bewegt sich der Bildschirm nach oben.

Nun bestehen zwei Möglichkeiten die Romroutine anzusprechen. Zum einen kann man eine kurze Maschinenroutine schreiben, in der man den Wert für das B-Register mittels eines Pokes ändert, oder man schreibt von vornherein zwei Routinen, eine für hoch, und eine für runter.
Wir haben uns für das letztere entschieden, da diese Methode unproblematischer ist und keine Änderungen Ihrerseits benötigt.

Wir müssen uns also klarmachen, was unsere Maschinenroutine bewirken soll.
Zuerst muß das B-Register mit einem Wert geladen werden. Dafür steht der Befehl LD B,N; zur Verfügung. Er hat den

Code 06. Der nun folgende Wert ist abhängig von der Bewegung, die der Bildschirm ausführen soll, für die Bewegung runter ist er null und für rauf 255. Dies ist im Prinzip der einzige Unterschied der beiden Routinen für die Bewegungsrichtungen.

Der nächste Schritt ist der Aufruf der Romroutine. Dies geschieht durch den Befehl CALL NN. Der Code dafür ist 205, ihm muß nun die Adresse folgen, sie lautet &BC4D. Diese muß in High- und Lowbyte aufgeschlüsselt werden. Dem Code 205 folgt also der Wert 77 (Lowbyte 4D), dann der Wert 188 (Highbyte BC).
Als letztes muß noch ein Rücksprung zum Basic erfolgen, dies geschieht durch ein Ret, der Code ist 201.

Die beiden Maschinenroutinen sehen folgendermaßen aus:

Für die Aufwärtsbewegung:

```
LD B,FF              06;255
CALL &BC4D           205;77;188
RET                  201
```

Für die Abwärtsbewegung:

```
LD B,0               06;0
CALL &BC4D           205;77;188
RET                  201
```

Das Basicprogramm zur Erzeugung der Maschinenroutinen legt diese nach 43880 (Hoch), bzw. nach 43886 (Runter) ab. Nach Beendigung des Programmes löschen Sie dieses mit DELETE -140, und können dann Ihr eigenes Programm schreiben, das die Routinen benutzt.

```
10 REM aufwaerts
20 MEMORY 43879
30 DATA 06,255,205,77,188,201
40 FOR adr=43880 TO 43885
50 READ wert
60 POKE adr,wert
70 NEXT
80 REM abwaerts
90 DATA 06,0,205,77,188,201
100 FOR adr=43886 TO 43891
110 READ wert
120 POKE adr,wert
130 NEXT adr
140 END
```

Erklärung zum Listing

10	Titel
20	Begrenzen des Basicspeichers
30	Datas für die Aufwärtsroutine
40	Eröffnen einer Schleife zum Einschreiben der Maschinensprachroutine
50	Lesen der Werte
60	Einschreiben der Werte in die Speicherstellen
70	Schließen der Schleife
80	Titel
90	Datas für die Abwärtsroutine
100-130	entsprechend 40-70
140	Programmende

Zur Benutzung der Routinen ist zu sagen, daß Sie diese nur mit CALL von basic aus ansprechen müssen (CALL 43880 für aufwärts, und CALL 43886 für abwärts).

Ein kleines Beispiel soll Ihnen die Möglichkeiten des SCROLL zeigen. Wir werden einen Schriftzug auf den Bildschirm bringen, und diesen mittels der beiden Routinen fünf Zeilen

nach oben und unten bewegen.
Sie können die Geschwindigkeit der Bewegung mittels Tastendruck bestimmen. Drücken Sie eine Taste, bewegt sich der Schriftzug, andernfalls wartet das Programm bis Sie wieder eine Taste drücken.

```
10 REM SCROLL Demo
20 MODE 1
30 LOCATE 15,12:PRINT"SCROLLDEMO"
35 REM aufwaerts
40 FOR n= 1 TO 5
50 IF INKEY$="" THEN 50
60 CALL 43880
70 NEXT n
75 REM aufwaerts
80 FOR n= 1 TO 5
90 IF INKEY$="" THEN 90
100 CALL 43886
110 NEXT n
120 END
```

Erklärung zum Listing

10	Titel
20	Einstellen des Bildschirmmodus
30	Ausgabe des Schriftzuges
35	Titel
40	Eröffnen der Schleife für die Anzahl der Bewegungen
50	Abfrage der Tastatur
60	Aufruf der SCROLL Routine (Aufwärts)
70	Schließen der Schleife
75	Titel
80	Eröffnen der Schleife für die Anzahl der Bewegungen
90	Abfrage der Tastatur
100	Aufruf der SCROLL Routine (Abwärts)
110	Schließen der Schleife
120	Programmende

Beachten Sie bitte, daß das Demo nicht läuft, wenn Sie nicht das Programm zur Erzeugung der Maschinenspracheroutinen laufen gelassen haben.

Zur Vereinfachung des Aufrufes der Routinen könnten Sie zum Beispiel zwei Variablen definieren und diesen die Werte für den Aufruf zuweisen. Die Variablen könnten so aussehen:
AUFWAERTS=43880 ABWAERTS=43886

Auf diese Art können Sie um einiges komfortabler mit den Routinen arbeiten.

Bei Benutzung der Routine ist zu beachten, daß der Farbwert der oben, bzw. unten hinzugefügten Zeile im A-Register steht. Von dieser Tatsache ausgehend, ist es möglich noch weitere Funktionen mit der SCROLL Routine zu erzeugen.

Wir haben ein kleines Programm geschrieben, das mit den SCROLL Routinen arbeitet und es zuläßt, die Richtung, die Länge der Bewegung, und die Farbe der eingefügten Zeile mittels einer Eingabe zu verändern.

Um diese zusätzlichen Funktionen zu erzeugen, müssen wir die Maschinenspracheroutine um einige Befehle erweitern. Die Richtung der Bewegung wird immer noch im B- Register gespeichert, jedoch benutzen wir das B- Register gleichzeitig als zählerfür die Anzahl der zuzufügenden Zeilen. Das A- Register beinhaltet den Farbwert der eingefügten Zeilen.
Da der Inhalt der Register bei einem Aufruf der Rom-Routine verlorengeht, müssen wir die Inhalte auf den Stapel legen, und sie nach dem Aufruf wieder zurückholen. Dies erreichen wir durch die Befehle PUSH (schieben) und POP (holen).

Die erweiterte Maschinenspracheroutine sieht nun so aus:

Adresse	Befehl	Code	Erklärung
43870	LD B,N	06	Anzahl der Zeilen (N)
43871		N	wird nach B geladen
43872	LD A,M	62	Farbwert (M) wird
43873		M	nach A geladen
43874	PUSH BC	197	B wird auf den Stapel gelegt
43875	PUSH AF	245	A wird auf den Stapel gelegt
43876	LD B,R	06	B wird mit dem Wert
43877		R	der Richtung geladen
43878	CALL &BC4D	205	Aufruf der Romroutine
43879		77	Lowbyte der Adresse
43880		188	Highbyte der Adresse
43881	POP AF	241	A wird vom Stapel genommen
43882	POP BC	193	B wird vom Stapel geholt
43883	DJNZ,e	16	folgt später
43884		245	
43885	RET	201	Rücksprung nach Basic

Um die Erklärung für den Befehl DJNZ,e zu geben, müssen wir etwas weiter ausholen. Zuerst die wörtliche Bedeutung: Decrementiere, springe um e, wenn nicht null.

Der Ablauf des Programmes benötigt einen Sprung zur Adresse 43874, damit die Anzahl der Zeilen nicht nur auf eins beschränkt ist. Das B-Register dient, wie schon erwähnt, zu Zählaufgaben. Es muß nun aber für jeden Durchlauf durch das Programm um eins erniedrigt werden. Dies geschieht durch das decrementiere. Ist das B- Register danach nicht gleich null, so springt das Programm zur Adresse 43874.

Es stellt sich nun aber die Frage, wie kommen wir von der Adresse 43874 auf den Wert 245.
Dies erklärt sich dadurch, daß das Programm um neun Bytes zurückspringen muß. Zu diesen neun werden zwei hinzugezählt, da der Programmcounter schon auf den nächsten Befehl zeigt. So ergibt sich eine Zahl von 11. Will man einen Rücksprung durchführen, so muß man von 256 die Sprungweite abziehen. In unserem Fall ergibt sich das Ergebnis 245 und dies ist der Wert der nach dem Befehlscode eingetragen werden muß.
Ist das B-Register gleich null, so kehrt das Programm nach Basic zurück.

```
10 REM SCROLLDEMO2
20 MEMORY 43869
30 adr =43869:scroll=43870
40 DATA 6,10,62,0,197,245,6,255,205,77
50 DATA 188,241,193,16,245,201
60 FOR co=1 TO 16
70 READ wert
80 POKE adr+co,wert
90 NEXT co
100 MODE 1
110 INPUT"Farbe";m
120 INPUT"Zeilenzahl";n
130 INPUT"aufwaerts(1) oder abwaerts(2)";r
140 IF r=1 THEN r=255 ELSE r=0
150 POKE 43873,m
160 POKE 43871,n
170 POKE 43877,r
180 CALL scroll
190 GOTO 100
```

Erklärung zum Listing:

10	Titel
20	Begrenzung des Basicspeicher
30	Festlegen der Programmanfangsadresse
40-50	Datas für die Maschinenroutine
60	Eröffnen der Schleife zum Einlesen der Datas
70	Einlesen der Maschinensprachcodes
80	Einschreiben der Werte in die Speicherstellen
90	Schließen der Schleife
100	Einstellen des Bildschirmmodus
110	Eingabe des Farbwertes
120	Eingabe der Zeilenzahl
130	Entscheidung, ob aufwärts oder abwärts
140	Verarbeiten der letzten Eingabe
150	Einschreiben des Farbwertes
160	Einschreiben der Zeilenzahl
170	Einschreiben der Bewegungsrichtung
180	Aufruf der Maschinensprachroutine
190	Rücksprung nach 100

Zur Benutzung des Programmes ist zu sagen, die Eingaben müssen in folgenden Bereichen liegen:

Farbwert= 0 bis 255
Zeilenzahl= 1 bis 25

Sollten Sie die Maschinenroutine in eins Ihrer Programme einbauen wollen, so benötigen Sie die folgenden Adressen:

43873	Der Farbwert muß in diese Speicherstelle eingeschrieben werden.
43871	Die Zeilenzahl muß in diese Speicherstelle eingeschrieben werden.
43877	Die Bewegungsrichtung muß in diese Speicherstelle eingeschrieben werden.
43870	Startadresse für die Maschinenspracheroutine

KAPITEL 7 ANWENDERPROGRAMME

7.1 EINLEITUNG

Als dieses Buch geschrieben wurde, war nach unserem Wissen noch keine (vernünftige) Software für den CPC - 464 vorhanden. Da ein Computer aber erst durch Programme "zum Leben erweckt" wird, liefern wir Ihnen nachfolgenden einige.

Wir haben besonderes Gewicht auf den CPC DATA gelegt, da ein Dateiverwaltungsprogramm das wichtigste, universellste und gebräuchlichste kommerzielle Programm ist. Natürlich eignet sich Ihr neuer Computer mit seinem Monitor und den 80 Zeichen ideal für solch ein Programm.

Die nachfolgenden Programme sind alle voll ablauffähig. Da sich aber inzwischen herumgesprochen hat, daß man nur durch Übung zum Meister wird, haben wir es uns "verkniffen" alle programmtechnischen Register zu ziehen. Dieses ist Ihre Sache. Das Basic, sowie das gute Handbuch, laden ja direkt zum selbst programmieren ein. Deshalb geben wir Ihnen bei der Erläuterungen der Programme Tips zur Verbesserung. So können Sie also selbst an dem Programm herumbasteln, es verbessern, schneller machen, eleganter u.s.w. .
Dabei wünschen wir Ihnen viel Spaß.
PROBIEREN GEHT ÜBER STUDIEREN.

7.2 DATEIVERWALTUNGSPROGRAMM

Ein Dateiverwaltungsprogramm, was ist das überhaupt? Eine Dateiverwaltung ist universell, im Gegensatz hierzu ist zum Beispiel eine Adressverwaltung oder eine Lagerverwaltung speziell. Sie sind nur für einen bestimmten Bereich zu gebrauchen. Eine Dateiverwaltung können Sie hingegen anpassen.

Dieses wird durch das Konzept einer Dateiverwaltung erreicht. Sie besteht aus DATENSÄTZEN und FELDERN. Zur Erläuterung dieser beiden Begriffe zieht man am besten den Ihnen sicherlich bekannten Komplex der Kartei heran. Eine Kartei besteht aus einer bestimmten Menge Karteikarten, auf denen eine bekannte, feste Anzahl von Eintragungen steht. Die ganze Kartei entspricht unserer DATEI, die Karteikarte entspricht unserem DATENSATZ und eine einzelne Eintragung entspräche einem FELD.
In einer Dateiverwaltung können Sie nun die Anzahl der Datensätze und Felder frei bestimmen. Jedem Feld können Sie nach Ihrem eigenen Wunsch, einen Namen geben. Wenn Sie also eine Adressdatei aufbauen möchten, würden Sie das erste Feld als NAMEN-VORNAMEN, das 2. als STRASSE, das 3. als ORT u.s.w. definieren. Schon wäre Ihre Dateiverwaltung als Adressdatei ausgerichtet. Das ist doch eine feine Sache? Jetzt verstehen Sie sicherlich, warum wir soviel Gewicht auf die Dateiverwaltung legen, mit etwas Fantasie, könnten Sie das Programm sogar zur Textverarbeitung benutzen.

Da das nachfolgende Programm etwas komplex und lang ist, haben wir es modular (also in einzelne, sinnvolle Teile) zerlegt. Wenn Sie das Programm abtippen, geben Sie natürlich alle Teile direkt hintereinander ein.

T E I L 1 - M E N Ü -

Im Menü werden Ihnen die Programmfunktionen angeboten, die das Programm zur Verfügung stellt. Sie erreichen den Programmteil, indem Sie die Nummer dieses Teiles eingeben, und danach die ENTER Taste drücken, das Programm verzweigt dann zum entsprechenden Teil. Haben Sie einen bestimmten Teil ausgeführt, springt das Programm immer wieder zum Menü zurück, so daß Sie einen anderen "Job" beginnen können.

```
10 REM******************************************************
*******************
20 REM*****************  C   O   P   Y   R   I   G   H   T **********
*******************
30 REM*****************     C P C 464    T E A M     **********
*******************
40 REM*****************  D A T A    B E C K E R     **********
*******************
50 REM*****************         1 9 8 4             **********
*******************
60 REM******************************************************
*******************
70 REM   D A T E I V E R W A L T U N G       V E R S I O N
  1 . 0 (6.6.)
80 REM******************************************************
*******************
90 MODE 2
100 LOCATE 20,3:PRINT"C    P    C        D    A    T    A"
110 LOCATE 20,5:PRINT STRING$(&24,"=")
120 LOCATE 12,7:PRINT"COPYRIGHT   1 9 8 4   by    D A T A
B E C K E R"
130 LOCATE 15,10:PRINT"                    M   E   N   U   E "
140 LOCATE 15,11:PRINT"                    ------------"
150 LOCATE 15,13:PRINT"DATEI erstellen         - 1 -"
160 LOCATE 15,14:PRINT"DATEI eingeben          - 2 -"
170 LOCATE 15,15:PRINT"DATEI pflegen           - 3 -"
180 LOCATE 15,16:PRINT"DATEI abspeichern       - 4 -"
190 LOCATE 15,17:PRINT"DATEI laden             - 5 -"
200 LOCATE 15,18:PRINT"      suchen            - 6 -"
210 LOCATE 15,19:PRINT"DATEI ausdrucken        - 7 -"
220 LOCATE 15,20:PRINT"      beenden           - 8 -"
230 LOCATE 15,21:PRINT"      loeschen          - 9 -"
240 PRINT:PRINT:INPUT"Ihre W A H L  bitte   (1-9)   >enter<"
;a
250 ON a GOSUB 280,630,1010,1440,1570,1700,1900,2020,2070

260 GOTO 90
```

Erläuterung zum Programm:

10-80 Hier befindet sich das Copyright sowie eine Angabe über die Version des Programms.

90 Hier wird mit einen CPC-Basic-Befehl auf den 80 Zeichen Modus umgeschaltet.

100-230 In diesen Zeilen wird mit Hilfe des Locate-Befehles das Menübild ausgegeben.

240 Hier wird nun verlangt, daß Sie den gewünschten Programmteil anwählen.

250 Dieses ist eine kurze Routine um die einzelenen Teile anzuspringen.

260 Wenn von einem Programmteil mittels RETURN zurückgesprungen wird, sorgt diese Zeile für einen Sprung zum Anfang des Menüs.

Verwendete Variablen:
----

a Enthält die Nummer des gewünschten Programmteils

Bedienung des Programmteils:

Die Bedienung des Menüs ist denkbar einfach. Sie geben nur die gewünschte Nummer des Programmteiles ein, und drücken die ENTER-Taste. Alles andere macht das Programm für Sie.

T E I L 2 - D A T E I E R S T E L L E N -

In diesem Teil des CPC-DATA müssen Sie die Datei definieren. Hierzu sind zwei Schritte nötig. Sie müssen erstens angeben, wie groß die Datei sein soll, und zweitens, wie die Felder der Datei benannt werden sollen. Nach der Definition haben Sie dann noch die Möglichkeit, Änderungen vorzunehmen für den Fall, daß Sie sich vertippt haben.

```
280 REM  d  a  t  e  i   e  r  s  t  e  l  l  e  n
290 CLS
300 PRINT STRING$(&50,"_")
310 LOCATE 15,3:PRINT"D  A  T  E  I    E  R  S  T  E  L  L
 E  N"
320 PRINT STRING$(&50,"_")
330 LOCATE 1,8:INPUT"Wieviele Datensaetze soll die Datei hab
en";datensaetze
340 PRINT:INPUT"Wieviele Felder soll ein Datensatz haben ";f
elder
360 DIM bezeichnungf$(felder)
370 REM  definieren der felder
380 CLS
390 PRINT"Defienieren der Felder eines Datensatzes"
400 hdatensaetze=datensaetze
410 hfelder=felder
420 PRINT"----------------------------------------"
430 PRINT:PRINT"(Sie haben ";felder;"Felder zu definieren)"
440 PRINT:PRINT
450 FOR felder=1 TO felder
460 PRINT"Bezeichnung des ";felder;"ten Feldes":INPUT"";beze
ichnungf$(felder)
470 NEXT felder
480 PRINT:INPUT"Moechten Sie Aenderungen vornehmen";aenderun
gen$
490 IF aenderungen$="n" GOTO 90
500 CLS
510 PRINT"Aendern von Bezeichnungen eines Feldes"
520 PRINT"--------------------------------------"
530 FOR felder=1 TO felder-1
540 PRINT"Bezeichnung";felder;":",bezeichnungf$(felder)
550 NEXT
560 PRINT:PRINT
570 INPUT"Welches Feld moechten Sie aendern";felder
580 PRINT"Bezeichnung des alten Feldes ";bezeichnungf$(felde
r)
590 INPUT"Neue Bezeichnung des  Feldes ";bezeichnungf$(felde
r)
600 INPUT"Moechten Sie noch eine Aenderung vornehmen";aender
ungen$
610 IF aenderungen$="n" GOTO 90 ELSE 560
620 RETURN
```

Erläuterung zum Programm:

290-320 Kopf des Programmteiles.
330+340 Eingabe der Größe der Datei.
350+360 "Bezeichnungs"- Variablen werden mit der gewünschten Größe dimensioniert.
400+410 Die Anzahl der Felder und Datensätze werden in Hilfsvariablen (hfelder, hdatensaetze) gerettet, da die alten Variablen durch Benutzung in Schleifen andere Werte erhalten können.
380-440 Kopf des Unterprogrammteils DEFINIEREN DER FELDER.
450-470 Schleife zum Einlesen der Bezeichnungen der Felder.
480 Eingabe, ob Änderungen vorgenommen werden sollen,
490 wenn "nein", dann Rücksprung zum Menü.
500-520 Kopf des Unterprogrammteils AENDERN.
530-550 Schleife zur Ausgabe der Bezeichnungen der Felder.
560-620 Routine zur Änderung einer Bezeichnung, wenn keine weitere Änderung erforderlich, erfolgt ein Rücksprung zum Menü.

Verwendete Variablen:
----

datensätze Enthält die Anzahl der Datensätze der Datei.
felder Enthält die Anzahl der Felder der Datei.
hdatensätze Enthält dauernd die Anzahl der Datensätze.
hfelder Enthält dauernd die Anzahl der Felder.
bezeichnungd$(datensätze) Enthält die Bezeichnung der Datensätze.
bezeichnungf$(felder) Enthält die Bezeichnung der Felder.
aenderungen$ Enthält eine ja-nein Entscheidung.

Verbesserungsmöglichkeiten:

Wie die Definition der Datei jetzt ausgelegt ist, liegt die Anzahl der Datensätze und der Felder fest, sie ist also statisch. Für den Fall, daß Ihre Datei größer wird als geplant, brauchen Sie aber ein dynamisches Konzept, das heißt, die Datei muß mitwachsen. Da dieses aber eine prinzipielle Veränderung ist, sollte sie nur von Fortgeschrittenen durchgeführt werden. Der Anfänger sollte z.B. versuchen, ein paar Zeilen zu schreiben, mit denen Fehleingaben verhindert werden, oder bei der Eingabe die WINDOW-Technologie verwenden. Schauen Sie hierzu einmal im Handbuch nach. Durch den Gebrauch von WINDOWS kann das "OUTFIT" sowie der Faktor Ergonomie entscheidend verbessert werden.

Bedienung des Programmteils:

Auch hier ist die Bedienung relativ einfach. Das Programm ist dialoggesteuert, d.h. Sie vollziehen mit dem Programm ein "Frage und Antwort"-Spiel. Zuerst geben Sie die Anzahl der Datensätze ein (z.B. bei einer Adressdatei, wieviele Personen Sie aufnehmen wollen) und dann die Anzahl der Felder (Wieviele Eintragungen Sie für eine Person benötigen). Danach geben Sie die Bezeichnung der Felder ein (Name, Straße, Ort,...).

Haben Sie diesen Vorgang abgeschlossen, können Sie wählen, ob Sie zum Menü zurück möchten (=nein eingeben) oder Änderungen vornehmen (=ja eingeben).

Möchten Sie Änderungen vornehmen, gibt Ihnen das Programm

zuerst die Bezeichnung der Felder aus. Sie müssen nun die Nummer, sowie die neue Bezeichnung angeben. Ist dieser Vorgang abgeschlossen, können Sie entweder noch eine Änderung vornehmen, oder das Programm springt endgültig zum Menü zurück.

TEIL 3 -DATEI EINGEBEN-

Der folgende Programmteil hat die Aufgabe, die definierte Datei mit Daten zu füllen. Sie können also nun Ihre Namen, Straßen, Orte usw. in die Datei einspeisen. Außerdem besteht die Möglichkeit, direkt hinterher einzelne Felder zu ändern. Am Anfang dieses Teiles haben Sie noch den Namen der Datei zu bestimmen, also zum Beispiel "Adressdatei", "Lagerverwaltung", "Alle meine Freundinnen" oder was Sie wünschen.

```
630 CLS
640 felder=hfelder
650 datensaetze=hdatensaetze
660 PRINT STRING$(&50,"_")
670 LOCATE 15,3:PRINT"D  A  T  E  I    E  I  N  G  E  B  E
 N"
680 PRINT STRING$(&50,"_")
690 LOCATE 1,10:PRINT"(Ihre Datei hat";datensaetze;"Datensae
tze und";felder;"Felder)"
700 PRINT:PRINT:INPUT"Wie soll die Datei heissen";dateiname$
710 CLS
720 DIM inhalt$(felder,datensaetze)
730 FOR datensaetze=1 TO datensaetze
740 felder=hfelder
750 FOR felder=1 TO felder
760 WINDOW SWAP 0
770 WINDOW 1,80,1,8
780 PRINT STRING$(&50,"-")
790 PRINT"Dateiname :";dateiname$
800 PRINT"Hoechstzahl Datensaetze :";hdatensaetze,"Anzahl Fe
lder :";hfelder
810 PRINT"Nummer des aktuellen Datensatzes :";datensaetze
820 PRINT"Nummer des aktuellen Feldes      :";felder
830 PRINT STRING$(&50,"-")
840 WINDOW 1,80,9,25
850 CLS
860 PRINT bezeichnungf$(felder);:INPUT"";inhalt$(felder,date
nsaetze)
870 NEXT felder
880 NEXT datensaetze
```

```
890 LOCATE 1,10
900 INPUT"Moechten Sie Aenderungen vornehmen";aenderungen$
910 IF aenderungen$="n" GOTO 90
920 MODE 2
930 PRINT"Aendern der Inhalte eines Feldes"
940 PRINT"--------------------------------"
950 PRINT:INPUT"Welches Feld moechten Sie aendern";felder
960 INPUT"In welchem Datensatz";datensaetze
970 PRINT"Inhalt des alten Feldes :";inhalt$(felder,datensae
tze)
980 INPUT"Neuer Inhalt des Feldes";inhalt$(felder,datensaetz
e)
990 INPUT"Moechten Sie noch eine Aenderung vornehmen";aender
ungen$
1000 IF aenderungen$="n" GOTO 90 ELSE GOTO 920
```

Erläuterung zum Programm:

630-690	Kopfbild des Programmteiles
650+660	Den Variablen "datensätze" und "felder" werden die alten Werte zugewiesen für den Fall, daß sie verändert wurden.
700	Hier wird der Name der Datei eingegeben.
720	Die Variable "inhalt$" wird dimensioniert, sie wird die Daten der Datei erhalten.
73o+75o	Die Schleifen zum Einlesen der Daten werden eröffnet.
770-830	Kopf des Eingabeteiles, dieser informiert Sie über Felder, Datensätze usw. Windowtechnologie
840-880	Hier werden jetzt die Daten Ihrer Datei eingelesen.
890-910	Entscheidung, ob Änderungen vorgenommen werden sollen, wenn "nein", Rücksprung zum Menü.
920-1000	Änderungsvorgang, mit abschließender Möglich keit zum weiteren Ändern.

Verwendete Variablen:

datensätze	Bekannt
felder	Bekannt
hdatensäze	Bekannt
hfelder	Bekannt
dateiname $	Enthält den Namen der Datei
inhalt$ (felder,datensätze)	Enthält alle Daten der Datei, zweidimensionale Array
änderungen$	Bekannt

Verbesserungsmöglichkeiten:

Prinzipiell gibt es noch zwei Verbesserungsmöglichkeiten. Die erste ist, dafür zu sorgen, daß bei der Eingabe eines Datensatzes nicht immer alle vorher eingegeben Felder gelöscht werden.

Die zweite ist, die Änderungroutine nach jedem Datensatz anzuspringen.

Dieses sind beides leichte Änderungen, die von jedem Anfänger durchgeführt werden können. Beachten Sie bitte den Gebrauch der Windows. Sie können diese Methode auf jeden Programmteil, natürlich auch auf jedes andere Programm, ausdehnen. Dieses hebt die Bedienerfreundlichkeit erheblich.

Bedienung des Programmteils:

Nachdem Sie den Teil EINGEBEN angewählt haben, werden Sie zuerst aufgefordert, den Namen der Datei einzugeben. Der

Name der Datei kann zum Beispiel die Art (Adressdatei, Lagerverwaltung) oder auch das Thema (alle meine Freundinnen, Geschäftspartner) sein.

Nun kommen Sie in das EINGABEBILD. Dort werden Sie in den oberen Zeilen ständig über den Namen, die Höchstzahl der Felder und Datensätze, sowie über aktuelle Feld- und Datensatznummer informiert. Im mittleren Teil geben Sie der Reihe nach (Feld für Feld) Ihre Daten ein. Dabei wird Feld und Datensatznummer dauernd fortgeschrieben.

Haben Sie Ihre Daten eingegeben, so haben Sie die Möglichkeit, Felder zu ändern. Alles was Sie zu tun haben, sagt das Programm Ihnen im Dialog. Möchten Sie eine weitere Änderung vornehmen, geben Sie bei der entsprechenden Frage ein "j" ein, andernfalls springt der Programmteil zum Menü zurück.

TEIL 4 -DATEI PFLEGEN-

Der nachfolgende Programmteil hat die Aufgabe, Ihre Datei immer "in Schuß" zu halten. Es kommt ja häufig vor, daß sich Daten ändern. Zum Beispiel, wenn ein Geschäftspartner von Ihnen umzieht, oder eine Bekannte heiratet und dann einen anderen Nachnamen hat. Oder vielleicht ist es nötig einen ganzen Datensatz zu löschen oder auch mehrere Felder. Diese Aufgabe übernimmt Programmteil 3.Man nennt diese Tätigkeit pflegen (oder ändern) einer Datei.

Um das Pflegen komfortabel zu gestalten, haben wir zwei Möglichkeiten eingebaut: Entweder geben Sie direkt die Stelle ein, an der das zu ändernde Datum (Datum = Einzahl von Daten) steht, oder Sie "blättern" (lassen sich jeden Datensatz einzeln zeigen) in der Datei und entscheiden sich dann, ob Sie Änderungen vornehmen wollen.

```
1010 CLS
1020 PRINT STRING$(&50,"_")
1030 LOCATE 15,3:PRINT"                    D A T E I       P F L E
 G E N "
1040 PRINT STRING$(&50,"_")
1050 PRINT:PRINT"Direktes aendern des Inhaltes eines Feldes
   - 1 -"
1060 PRINT"Blaettern mit nachfolgendem aendern              - 2
 -"
1070 PRINT:INPUT"Bitte waehlen Sie ";a
1080 IF a=2 GOTO 1170
1090 CLS
1100 PRINT"Direktes aendern des Inhaltes eines Feldes"
1110 PRINT:PRINT:PRINT:INPUT"Welches Feld moechten Sie aende
rn ";felder
1120 INPUT"In welchem Datensatz ";datensaetze
1130 PRINT:PRINT:PRINT"Inhalt des alten Feldes :";inhalt$(fe
lder,datensaetze)
1140 INPUT"Neuer Inhalt des Feldes  ";inhalt$(felder,datensa
etze)
1150 PRINT:PRINT:INPUT"Moechten Sie noch eine Aenderung vorn
ehmen ";aenderungen$
1160 IF aenderungen$="j" GOTO 1090 ELSE RETURN
```

```
1170 CLS
1180 felder=hfelder
1190 datensaetze=hdatensaetze
1200 FOR datensaetze=1 TO datensaetze
1210 WINDOW 1,80,1,8
1220 PRINT STRING$(&50,"-")
1230 PRINT"Aktuelle Datei :"dateiname$
1240 PRINT"Aktueller Datensatz :"datensaetze
1250 PRINT STRING$(&50,"-")
1260 GOSUB 1290
1270 INPUT"Moechten Sie Aenderungen vornehmen";aenderungen$:
IF aenderungen$="n" THEN 1420
1280 GOTO 1350
1290 WINDOW 1,80,9,20
1300 CLS
1310 FOR felder = 1 TO hfelder
1320 PRINT bezeichnungf$(felder);":";inhalt$(felder,datensae
tze)
1330 NEXT felder
1340 RETURN
1350 WINDOW 1,80,21,25
1360 CLS
1370 PRINT STRING$(&50,"-")
1380 INPUT"Welches Feld moechten Sie aendern ";felder
1390 INPUT"Neuer Inhalt des Feldes ";inhalt$(felder,datensae
tze)
1400 INPUT"Moechten Sie noch eine Aenderung vornehmen ";aend
erungen$
1410 IF aenderungen$="j" GOTO 1360
1420 NEXT datensaetze
1430 RETURN
```

Erläuterung zum Programm:

1010-1060	Titelbild des Programmteilkopfes.
1070	Wahl des Unterprogrammteils.
1080	Sprung zum entsprechenden Unterprogrammteil.
1090-1140	Änderungsvorgang durch Angabe der Datensatznummer und der Feldnummer des zu ändernden Datums.
1150+1160	Entscheidung, ob Sie noch ein Datum ändern möchten, wenn "nein", Rücksprung zum Menü, sonst Sprung zum Anfang des Unterprogrammteils
1170-1250	Kopf des anderen Unterprogrammteils.
1200	Eröffnung der Schleife für die Anzeige der Datensätze.

1260	Sprung zur Routine, die die Felder anzeigt.
1290-1330	Routine, die mittels einer Schleife die Bezeichnung, sowie den Inhalt der Felder des Datensatzes ausgibt.
1340	Rücksprung nach 1270.
1270	Entscheidung, ob etwas geändert worden soll, wenn "nein", dann nächster Datensatz, bei "ja", Sprung nach 1350.
1350-1400	Änderungsroutine.
1400+1410	Entscheidung, ob noch ein Feld im Datensatz geändert werden soll, sonst nächster Datensatz.

Verwendete Variablen:
----

a	Enthält den Wert, welcher Unterprogrammteil angesprungen werden soll
felder	Bekannt
datensaetze	Bekannt
hfelder	bekannt
hdatensaetze	Bekannt
inhalt$(felder,datensaetze)	Bekannt
äenderungen$	Bekannt
dateiname$	Bekannt

Verbesserungsmöglichkeiten:

Verbesserungen sind eigentlich nur im zweiten Teil nötig. Dort ist es im Augenblick nicht möglich, durch Tastendruck ins Menü zurückzuspringen, Sie müssen erst alle Datensätze durchblättern. Da dieses aber bei einer großen Datei sehr lästig sein kann, wenn Sie Ihren zu ändernden Datensatz schon längst gefunden haben, sollte hier Abhilfe geschaffen

werden. Dieses können Sie durch eine kurze Routine bewerkstelligen, die überprüft, ob eine bestimmte Taste, die für den Rücksprung ins Menü vorgesehen ist (zum Beispiel :&), gedrückt ist. Ist diese Taste gedrückt, erfolgt mit einem einfachen GOTO-Befehl ein Rücksprung zum Menü.

Bedienung des Programmteiles:

Zuerst haben Sie die Entscheidung zu treffen, ob Sie direkt ändern möchten, oder erst blättern wollen. Direkt ändern können Sie nur, wenn Feld- und Datensatznummer des Datums bekannt sind. Sonst müssen Sie Teil 2 anwählen.

Haben Sie Teil 1 angewählt, geben Sie zuerst die Feldnummer und dann die Datensatznummer ein. Darauf gibt Ihnen das Programm den alten Feldinhalt aus und Sie haben die Möglichkeit, diesen zu ändern. Danach können Sie wählen, ob Sie noch ein Feld ändern möchten. Wenn "nein", springt das Programm zum Menü zurück.

Haben Sie Teil 2 angewählt, gibt Ihnen das Programm die Felder des ersten Datensatzes aus. Danach haben Sie die Möglichkeit, diese zu ändern. Haben Sie Ändern gewählt, müssen Sie nur noch die Feldnummer und den neuen Inhalt eingeben. Jetzt können Sie entweder noch ein Feld ändern, oder sich den nächsten Datensatz ansehen.

TEIL 5 -DATEI ABSPEICHERN-

Der folgende Programmmteil hat die Aufgabe, die von Ihnen mit dem CPC-DATA erstellten Dateien auf Band abzuspeichern, damit Ihnen diese beim Ausschalten des Computers nicht verloren gehen.

```
1440 CLS
1450 PRINT STRING$(&50,"_")
1460 PRINT"            D  A  T  E  I        A  B  S  P  E  I
  C  H  E  R  N"
1470 PRINT STRING$(&50,"_")
1480 LOCATE 5,10:PRINT"Ihre Datei wird abgespeichert,bitte w
arten Sie."
1490 OPENOUT dateiname$
1500 FOR datensaetze=1 TO hdatensaetze
1510 FOR felder=1 TO hfelder
1520 PRINT#9,inhalt$(felder,datensaetze)
1530 NEXT felder
1540 NEXT datensaetze
1550 CLOSEOUT
1560 RETURN
```

Erläuterung zum Programm:

1440-1470	Kopf des Programmteiles
1480	Mitteilung für den Benutzer
1490	Eröffnung einer Datei auf Band unter dem Namen, der im Computer befindlichen Datei
1500-1540	Schleife, um die Daten auf Band zu schreiben
1550	Schließen der Datei
1560	Rücksprung zum Menü

Verwendete Variablen:
----

dateiname$	Bekannt
datensätze	Bekannt
felder	Bekannt
hdatensätze	Bekannt
hfelder	Bekannt
inhalt$(felder, datensätze)	Bekannt

Verbesserungsmöglichkeiten:

Prinzipiell keine Verbesserungen. Bei Bedarf ist es möglich, "Outfit" (also das Aussehen oder die Ergonomie am Bildschirm) des Programms verbessern.

Bedienung des Programmteiles:

Nachdem Sie den Programmteil angewählt haben, erscheint nach kurzer Zeit die Aufforderung die Tasten PLAY und RECORD auf Ihrem Datenrecorder zu drücken. Haben Sie dieses getan, betätigen Sie nur noch eine beliebige Taste. Den Rest erledigt das Programm für Sie.

TEIL 6 -DATEI LADEN-

Der nachfolgende Teil hat die Aufgabe, Dateien, die Sie mit Ihrem CPC erstellt und dann abgespeichert haben, wieder einzuladen. Hierzu muß allerdings die Anzahl der Felder und der Datensätze der Datei bekannt sein. Diese Parameter können Sie im Programmteil 1 einstellen.

```
1570 CLS
1580 PRINT STRING$(&50,"_")
1590 PRINT"                    D  A  T  E  I          L  A
D  E  N"
1600 PRINT STRING$(&50,"_")
1610 LOCATE 5,10:PRINT"Ihre Datei wird geladen,bitte warten
Sie."
1620 OPENIN "
1630 FOR datensaetze=1 TO hdatensaetze
1640 FOR felder=1 TO hfelder
1650 INPUT#9,inhalt$(felder,datensaetze)
1660 NEXT felder
1670 NEXT datensaetze
1680 CLOSEIN
1690 RETURN
```

Erläuterung des Programmteiles:

1570-1600	Ausgabe des Programmteilkopfes
1610	Mitteilung für den Benutzer
1620	Eröffnung einer Datei zum Einlesen von Kassette
1630-1670	Schleife zum Einlesen der Datei von Kassette
1680	Schließen der Datei
1690	Rücksprung zum Menü

Verwendete Variablen:
----

datensätze	Bekannt
felder	Bekannt
hdatensätze	Bekannt
hfelder	Bekannt
inhalt$(felder, datensätze)	Bekannt

Verbesserungen:

Wie beim Programmteil DATEI ABSPEICHERN, sind Veränderungen im optischen Bereich möglich. Außerdem wäre es möglich, die Einleseroutine so zu verändern, daß die Anzahl der Felder und Datensätze nicht bekannt sein muß.

Bedienung des Programmteiles:

Die Bedienung ist wieder denkbar einfach. Nachdem Sie diesen Teil angewählt haben, spulen Sie die Kassette vor die Datei, die Sie laden möchten. Dann drücken Sie nach der Aufforderung durch das Programm die Taste PLAY und danach noch irgend eine andere Taste. Der Programmteil liest dann automatisch die Datei von Band.
Danach erfolgt ein Rücksprung ins Menü.

T E I L 7 - S U C H E N -

Der folgende Programmteil bietet Ihnen die Möglichkeit, in Ihrer Datei ein bestimmtes Datum zu suchen, also zum Beispiel einen Namen, eine Telefonnummer oder ähnliches. Dieses geschieht einfach, indem Sie das zu suchende Datum eingeben und wieviele Stellen signifikant, das heißt, von Bedeutung sein sollen. Eine kleine Anzahl von signifikanten Stellen hat den Vorteil, daß der Begriff schneller gefunden wird. Allerdings wurde eine relativ schnelle Suchroutine verwandt (für unsortierte Listen). Die Zahl muß nicht allzu klein gewählt werden.
Haben Sie zum Beispiel nur 3 Stellen gewählt, sind für den Computer "Schulz" und "Schmitz" dieselben Personen, da die ersten drei Stellen übereinstimmen. Wählen Sie die Zahl der signifikanten Stellen mit vier, würde dieses nicht passieren.

```
1700 CLS
1710 PRINT STRING$(&50,"_")
1720 PRINT"                              S   U   C   H   E   N"
1730 PRINT STRING$(&50,"_")
1740 LOCATE 5,10:INPUT"Wie heisst der zu suchende Begriff";b
egriff$
1750 LOCATE 5,11:INPUT"Wieviele signifikante Stellen     ";s
tellen
1760 PRINT:PRINT:PRINT"BITTE WARTEN,ES WIRD GESUCHT"
1770      PRINT"============================"
1780 SUCH$=LEFT$(BEGRIFF$,STELLEN)
1790 FOR DATENSAETZE=1 TO hdatensaetze
1800 FOR felder=1 TO hfelder
1810 hilf$=inhalt$(felder,datensaetze)
1820 find$=LEFT$(hilf$,stellen)
1830 IF such$=find$ THEN GOTO 1880
1840 NEXT felder
1850 NEXT datensaetze
1860 INPUT"Begriff nicht gefunden,noch einen Begriff suchen
";wahl$
1870 IF wahl$="n" RETURN ELSE GOTO 1700
1880 PRINT"Der gesuchte Begriff ";begriff$;" steht im Feld "
;felder;" im Datensatz ";datensaetze:INPUT"  Noch einmal suc
hen ";wahl$
1890 IF wahl$="n" THEN RETURN ELSE GOTO 1700
```

Erläuterungen zum Programm:

1700-1730	Kopf des Programmteiles.
1740-1750	Einlesen der Informationen über den zu suchenden Begriff.
1760-1770	Information für den Benutzer.
1780	der Variablen such$ werden die signifikanten Stellen des zu suchenden Begriffes zugewiesen.
1790-1800	die Schleife für den Suchvorgang wird eröffnet.
1810	der Variablen hilf$ werden mittels der Schleife nacheinander die Inhalte der Datei zugewiesen.
1820	Der Variablen find$ werden die signifikanten Stellen der Inhalte der Datei nacheinander zugewiesen.
1830	Wenn such$=find$ ist, wurde der zu suchende Begriff gefunden.
1840+1850	Ende der beiden Schleifen.
1860+1870	Verzweigte sich das Programm während der Schleife nicht, wurde der Begriff nicht gefunden, entsprechende Meldung wird in dieser Zeile ausgegeben. Es besteht die Möglichkeit, einen neuen Begriff zu suchen, sonst Rücksprung ins Menü
1880+1890	Hierhin wird verzweigt, wenn der Begriff gefunden wurde. Es wird der Begriff selbst, sowie Datensatz- und Feldnummer ausgegeben, außerdem besteht die Möglichkeit, noch einmal zu suchen. Sonst Rücksprung ins Menü.

Verwendete Variablen:
---

begriff$	Enthält den zu suchenden Begriff
stellen	Enthält die Anzahl der signifikanten Stellen
such$	Enthält die signifikanten Stellen des zu suchenden Begriffs
hilf$	Hifsvariable, der nacheinander die Inhalte der Datei zugewiesen werden
find$	Enthält die signifikanten Stellen des Datums der Datei mit dem such$ verglichen werden soll
datensätze	Bekannt
felder	Bekannt
hdatensaetze	Bekannt
hfelder	Bekannt
wahl$	Bekannt

Verbesserungsmöglichkeiten:

An der Suchroutine selbst brauchen Sie nichts zu verändern. Sie ist optimal auf den CPC abgestimmt, und für eine Basicroutine selbst bei größeren Datenmengen sehr schnell. Die einzige Verbesserungmöglichkeit, die Sie noch haben, wäre, anstatt nur Datensatz- und Feldnummer auszugeben, gleich den gesamten Datensatz mit allen Feldern auszugeben. Dieses dürfte auch für den Anfänger kein Problem darstellen, da ja Datensatznummer und die Anzahl der Felder in einem Datensatz (Variable hfelder) bekannt sind. Somit läßt sich diese Verbesserung mit einer einzelnen Ausgabeschleife gut realisieren.

Bedienung des Programmteiles:

Nach Anwahl von SUCHEN müssen Sie zuerst den zu suchenden Begriff eingeben. Dann werden Sie aufgefordert, die Anzahl der signifikanten Stellen einzugeben. Was diese für eine Bedeutung haben, erfuhren Sie in der Einleitung zu diesem Teil. Lesen Sie dort ruhig noch einmal nach, wenn Sie es nicht mehr wissen sollten.
Wurde der Begriff gefunden, gibt Ihnen das Programm die Datensatz- und Feldnummer aus. Danach müssen Sie sich entscheiden, ob Sie noch etwas suchen möchten (Taste "j"), sonst springt das Programm zum Menü zurück.
Wurde der Begriff nicht gefunden, teilt Ihnen das Programm dieses ebenfalls mit. Sie können jetzt ins Menü zurück oder noch einen Suchvorgang einleiten. Sie sollten dieses ruhig noch einmal mit demselben Begriff, aber weniger signifikanten Stellen tun, da es öfters vorkommt, daß man sich vertippt, und solche Fehler kann dieses Programm nicht erkennen. Hierfür gibt es sogenannte "Toleranzroutinen", die zum Beispiel bei 90%tiger Übereinstimmung einen Begriff ausgeben. Derartige Routinen finden sich aber meist nur in nicht ganz preiwerten Dateiverwaltungen für Business-Maschinen.

T E I L 8 - D A T E I A U S D R U C K E N -

Der folgende Programmteil gibt Ihnen die Möglichkeit, Ihre mit dem CPC-DATA erstellten Dateien auf einem angeschlossenen Drucker auszugeben.
Böse Zungen sagen ja, daß mit dem Computer erst das Zeitalter des Papiers begonnenn hätte, aber Realität ist, daß die meisten Computerbenutzer alles auch gerne schwarz auf weiß haben möchten. Dieses ist nicht nur ein Hang zum Althergebrachten, sondern hat auch praktische Seiten. Wenn zum Beispiel einmal Ihre Datei verlorengeht (Sie können sie nicht mehr laden oder haben sie überschrieben), dann haben Sie die Daten wenigstens noch auf dem Papier vorliegen.

```
1900 CLS
1910 PRINT STRING$(&50,"_")
1920 PRINT"                D  A  T  E  I   a  u  s  d  r  u  c
  k  e  n"
1930 PRINT STRING$(&50,"_")
1940 LOCATE 5,10:PRINT"Datei wird ausgedruckt.Bitte warten."

1950 FOR datensaetze=1 TO hdatensaetze
1960 FOR felder=1 TO hfelder
1970 PRINT#8,inhalt$(felder,datensaetze)
1980 NEXT felder
1990 NEXT datensaetze
2000 PRINT:INPUT"Noch einen Ausdruck";wahl$
2010 IF wahl$="j" GOTO 1950 ELSE RETURN
```

Erläuterung zum Programm:

1900-1930	Kopf des Programmteiles.
1940	Information für den Benutzer.
1950-1990	Routine zum Ausdruck mittels einer Schleife, dem PRINT-Befehl und der Adresse des Druckers (8).
2000-2010	Möglichkeit, noch einen Ausdruck anfertigen zu lassen, wenn "n" gedrückt wurde, erfolgt ein Rücksprung ins Menü.

Verwendete Variablen:
----

datensätze	Bekannt
felder	Bekannt
hdatensätze	Bekannt
hfelder	Bekannt
wahl$	Bekannt

Verbesserungsmöglichkeiten:

Die jetzige Routine zum Ausdruck ist eine relativ einfache, eine sogenannte unformatierte Ausgabe. Die einzelnen Felder der Datei werden einfach der Reihe nach ausgegeben.

Im Gegensatz hierzu steht eine sogenannte formatierte Ausgabe. Solch eine Ausgabe könnte zum Beispiel Datensatz- und Feldnummer, Feldbezeichnung u.s.w. mit ausgeben außerdem Absätze zwischen den Datensätzen lassen und vieles mehr. Alle diese Erweiterungen lassen sich mit de hervorragenden Befehl PRINT USING Ihres CPC realisieren. Lesen Sie sich diesen Abschnitt im Handbuch sorgfältig durch, er ist zwar nicht ganz einfach, kann aber viel Arbeit mit Stringfunktionen (right$, ,left$ ...) ersparen.

Bedienung des Programmteiles:

Die Bedienung dieses Programmteiles ist wieder spielend einfach. Nach dem Anwählen des Programmteils wird die Datei ausgedruckt. Sie brauchen nur noch am Ende eines Ausdruckes wählen, ob Sie Ihre Datei noch einmal zu Papier bringen wollen.

T E I L 9 - B E E N D E N D E S P R O G R A M M S -

Der nachfolgende Programmteil hat nun die Aufgabe, das Programm abzubrechen. Möchten Sie nicht mehr mit Ihrem CPC-DATA arbeiten, wählen Sie diesen Teil an. Das Programm beendet sich selbst.

```
2020 CLS
2030 PRINT STRING$(&50,"_")
2040 PRINT"    B  E  E  N  D  E  N    D  E  S    P  R  O
 G  R  A  M  M  E  S"
2050 PRINT STRING$(&50,"_")
2060 LOCATE 5,10:INPUT"Moechten Sie das Programm wirklich be
enden";b$:IF b$ = "j" THEN END ELSE RETURN
```

Erläuterungen zum Programm:

2020-2050	Ausgabe des Programmteilkopfes.
2060	Entscheidung, ob Programm wirklich beendet werden soll, wenn "nein", erfolgt ein Rücksprung ins Menü.

Verwendete Variablen:

b$	Enthält die ja/nein-Entscheidung, ob das Programm beendet werden soll

Verbesserungsmöglichkeiten:

Verbesserungsmöglichkeiten sind nur optischer Natur, da der Teil programmtechnisch keine Anforderungen stellt.

Bedienung des Programmteiles:

Nach Anwahl haben Sie sich nur zu entscheiden, ob Sie das Programm wirklich beenden möchten. Diese Rückfrage wurde eingebaut, da es ja vorkommen kann, daß Sie diesen Programmteil aus Versehen anwählen und dann wären alle Ihre nicht abgespeicherten oder ausgedruckten Daten verloren.

TEIL 10 -LÖSCHEN-

Der letzte Programmteil gibt Ihnen die Möglichkeit, Daten zu löschen. Dieses wird zum Beispiel nötig, wenn ein Bekannter verzogen oder verstorben ist, Sie einen Artikel nicht mehr führen oder ähnliches. Sie sollten allerdings vorsichtig mit dem Löschen sein, da Informationen schnell gelöscht, aber schwer wieder bechafft werden können.

```
2070 CLS
2080 PRINT STRING$(&50,"_")
2090 PRINT"                         L  O  E  S  C  H  E  N"
2100 PRINT STRING$(&50,"_")
2110 LOCATE 10,10:PRINT"ALLES     loeschen      - 1 -"
2120 LOCATE 10,11:PRINT"DATENSATZ loeschen      - 2 -"
2130 LOCATE 10,14:INPUT"Bitte waehlen Sie ";wahl$
2140 IF wahl$="2" GOTO 2210
2150 CLS
2160 PRINT"ALLES loeschen"
2170 PRINT"--------------"
2180 LOCATE 5,10:INPUT"Moechten Sie wirklich alles loeschen
";wahl$
2190 IF wahl$="j" THEN CLEAR
2200 GOTO 90
2210 CLS
2220 PRINT"Datensatz loeschen"
2230 PRINT"------------------"
2240 LOCATE 5,10:INPUT"Welchen Datensatz moechten Sie loesch
en ";datensaetze
2250 FOR felder=1 TO hfelder
2260 inhalt$(felder,datensaetze)="    "
2270 NEXT felder
2280 LOCATE 5,15:INPUT"Moechten Sie noch einen Datensatz loe
schen ";wahl$
2290 IF wahl$="j" GOTO 2210 ELSE RETURN
```

Erklärung zum Programm:

2070-2120	Kopf des Programmteiles
2130+2140	Entscheidung für einen der beiden Programmteile
2150-2200	Routine zum Löschen aller Daten mittels des Basic-Befehls CLEAR, vorher wird nachgefragt, ob wirklich alle Daten gelöscht werden sollen
2220-2230	Kopf des Unterprogrammteils
2240	Eingabe, welcher Datensatz gelöscht werden soll
2250	Eröffnung einer Schleife für die Felder
2260	Löschen der Daten durch Zuweisung eines Leerstrings.
2270	Schleifenende.
2280-2290	Möglichkeit der Wahl ob noch ein Datensatz gelöscht werden soll, wenn "nein", erfolgt ein Rücksprung zum Menü.

Verwendete Variablen:
----

wahl$	Bekannt
datensätze	Bekannt
felder	Bekannt
hfelder	Bekannt
inhalt$(felder, datensätze)	Bekannt

Verbesserungmöglichkeiten:

Im Unterprogrammteil ALLES löschen sind Verbesserungen wieder nur optischer Natur. Im Teil Datensatz löschen, ließe sich noch eine Erweiterung vornehmen. Diese wäre selektiv

nur ein Feld eines Datensatzes zu löschen. Dieses läßt sich einfach bewerkstelligen, indem die Feldnummer abgefragt wird, und dann der mit Feld- und Datensatznummer festgelegten Variable inhalt$ ein Leerstring zugewiesen wird.

Bedienung des Programmteiles:

Nachdem Sie den Teil angewählt haben, müssen Sie sich entscheiden, ob Sie alle Daten löschen möchten, oder nur einen einzelnen Datensatz.

Haben Sie sich für ALLES löschen wirklich entschieden, fragt das Programm noch einmal nach, ob Sie sich sicher sind. Wenn Sie Ihre Daten nicht löschen möchten, geben Sie ein "n" ein, sonst ein "j".

Haben Sie sich für den zweiten Teil entschieden, müssen Sie die Nummer des Datensatzes eingeben, den Sie löschen möchten. Das Löschen erledigt das Programm für Sie. Ist der Datensatz gelöscht, haben Sie die Möglichkeit noch einen weiteren Datensatz zu löschen. Möchten Sie dieses nicht, geben Sie ein "n" ein und das Programm springt zum Menü zurück.

TEIL 11 -SCHLUSSWORT-

Sie haben jetzt einige Seiten dieses Buches durchgearbeitet, und ca. 9k Programm eingegeben.
Wir haben Ihnen eine vollständige Dateiverwaltung gegeben, zusätzlich alles, was an Dokumentation nötig ist.
Mit diesen Informationen sollte es Ihnen möglich sein, das Programm vollständig, von der Seite der Bedienung, aber auch von der Seite der Programmierung, zu verstehen. Wir haben die Struktur so einfach wie möglich gehalten, damit Sie alles nachvollziehen können. Arbeiten Sie das Programm sowie die Erklärungen genau durch. Auf diese Weise können Sie eine Menge über Programmieren lernen.
Sollte es Ihnen dann noch gelingen, die angegebenen Verbessserungsmöglichkeiten in das Programm einzubauen, haben Sie nicht nur eine erstklassige Dateiverwaltung, sondern auch eine Menge mehr an Programmierkenntnissen. Also, arbeiten Sie mit dem Programm, es kann Ihnen nur Vorteile bringen.

7.3 TEXTVERARBEITUNG

Um Ihren CPC nicht nur als Hilfsmittel zur Beschäftigung mit der Informatik zu gebrauchen, sondern auch als vollwertiges Arbeitsmittel, liefern wir Ihnen nachfolgend das zweite Programm mit kommerziellem Hintergrund, eine Textverarbeitung.

Welche Augaben hat eine solche Textverarbeitung? Eine Textverarbeitung soll alle Arbeiten rationalisieren, die im Bereich des geschriebenen Wortes angesiedelt sind. Sie können zum Beispiel einen Text, den Sie einmal erstellt haben, auf Kassette abspeichern und später immer wieder einladen und ausdrucken.Natürlich ist es auch möglich, einen Tippfehler, den Sie auf dem Bildschirm gemacht haben, ohne sichtbare Spuren zu berichtigen. Wie eine Änderung in einem Brief aussieht, der mit Schreibmaschine geschrieben wurde, wissen Sie sicherlich aus eigener Erfahrung.

Es würde den Rahmen dieses Buches sprengen, wenn wir hier alle Vorteile einer Textverarbeitung genauestens erläutern würden. Es ist aber auch nicht nötig. Viele werden Sie bei der Arbeit mit unserem CPC-Text kennenlernen, über die restlichen werden Sie sich relativ genaue Vorstellungen machen können, wenn Sie erst einmal eine Zeit mit dem Programm gearbeitet haben.

Wir legen Wert darauf, das Programm so komfortabel wie möglich zu gestalten. Natürlich wissen wir, daß es bessere gibt (so zum Beispiel den Textomat von DATA BECKER für den COMMODORE 64, auf dem dieses Buch erstellt wurde), aber trotzdem sind wir auf unser Programm stolz.

Besonders gut ist der Punkt "TEXT eingeben" gelungen, da es möglich ist, Text völlig frei an jeder Stelle des Bildschirms einzugeben, etwas, was normalerweise bei einem Basicprogramm dieser Länge nicht üblich ist.

Wie beim CPC-DATA haben wir die Erklärung möglichst ausführlich gestaltet, damit Sie zum einem mit dem Programm richtig arbeiten können und zum anderen, damit Sie wieder die Möglichkeit haben, Teile anders zu programmieren.

T E I L 1 - M E N Ü -

Das Menü des CPC-TEXTES hat dieselbe Aufgabe, wie das Menü des CPC-DATAS.

Sollten Ihnen nicht mehr die genaue Bedeutung eines Menüs bekannt sein, schauen Sie ruhig noch einmal beim Kapitel 7.2 nach.

Wir haben wieder die Menütechnologie gewählt, weil diese für den Anfänger am einfachsten ist. Um Ihnen dieses verständlich zu machen, möchten wir noch einen kurzen Ausflug in die Welt der sogenannten Textverarbeitungsphilosophie unternehmen.

Es gibt im wesentlichen zwei Arten, wie eine Textverarbeitung bedient werden kann. Dieses ist erstens die Menütechnologie und zweitens die Befehlstechnologie.

Die Menütechnologie ist Ihnen bekannt. Sie haben immer ein Menü, zu dem jeder Programmteil nach dem Abarbeiten zurückspringt, und von dem Sie jeden Teil erreichen können.

Anders bei der Befehlstechnologie. Bei dies Textverarbeitungsphilosophie müssen Sie ständig eine große Zahl von Befehlen im Kopf behalten, da Sie nur so Funktionen des Programms auslösen können. Dieses ist sehr mühsam, da man eine nicht unbeträchtliche Zeit braucht, bis man alle Befehle im Kopf hat. Fällt einem dann ein bestimmter Befehl nicht ein, muß man erst lange im Handbuch suchen.

Sicherlich wissen Sie nun, warum unser Programm menügesteuert ist.

```
10 -250
20 REM****************** C  O  P  Y  R  I  G  H  T *********
*******************
30 REM******************      1  9  8  4          *********
*******************
40 REM******************     CPC 464  TEAM        *********
*******************
50 REM******************   D A T A   B E C K E R   *********
*******************
60 REM**************************************************
*******************
70 REM* T E X T V E R A R B E I T U N G      V E R S I O N
   1  .  0        *
80 REM**************************************************
*******************
90 DIM inhalt$(82,25)
100 MODE 2
110 LOCATE 26,3:PRINT" C  P  C      T  E  X  T"
120 LOCATE 20,5:PRINT STRING$(&24,"=")
130 LOCATE 12,7:PRINT"COPYRIGHT    1984   BY    D A T A
B E C K E R"
140 LOCATE 30,10:PRINT"M  E  N  U  E"
150 LOCATE 30,11:PRINT"-------------"
160 LOCATE 20,12:PRINT"TEXT erstellen          - 1 -"
170 LOCATE 20,13:PRINT"TEXT ansehen            - 2 -"
180 LOCATE 20,14:PRINT"TEXT aendern            - 3 -"
190 LOCATE 20,15:PRINT"TEXT loeschen           - 4 -"
200 LOCATE 20,16:PRINT"TEXT abspeichern        - 5 -"
210 LOCATE 20,17:PRINT"TEXT laden              - 6 -"
220 LOCATE 20,18:PRINT"TEXT drucken            - 7 -"
230 LOCATE 20,19:PRINT"PROGRAMM beenden        - 8 -"
240 LOCATE 5,22:INPUT"Ihre  W A H L  bitte    (1-8)   >enter
<";a
250 ON a GOTO 260,680,850,1010,1080,1240,1390,1540
```

Erläuterung zum Programmteil:

10-80	Ausgabe des Copyright.
90	Dimensionierung der Variablen, die den Text enthalten soll.
100	Einstellen des 80 - Zeichen Modus.
110-230	Ausgabe des Menübildes.
240	Anwahl des Programmpunktes.
250	Ansprung des Programmteils, der vorher angewählt wurde.

Verwendete Variablen:
----

x — Enthält die Nummer des angewählten Programmteiles

Verbesserungsmöglichkeiten:

Sie sind in diesem Teil nur optischer Natur, da programmtechnisch keine Anforderungen gestellt werden.

Bedienung des Programmteils:

Die Bedienung erfolgt wie beim CPC-DATA. Wenn Sie aufgefordert werden, "Ihre Wahl" zu treffen, brauchen Sie nur die Nummer des Teiles einzugeben, mit dem Sie arbeiten möchten. Den Rest erledigt das Programm für Sie.

T E I L 2   - T E X T   E R S T E L L E N -

Der nachfolgende Programmteil hat die Aufgabe, den Text einzulesen.
Sie können Ihren Text frei eingeben, indem Sie einfach wie bei einer Schreibmaschine die Tasten der Computertastatur drücken. Dabei zeigt Ihnen ein "Cursor" (ein kleiner Punkt) an, wo das nächste Zeichen ausgegeben wird. Diesen Punkt können Sie mit Hilfe der Cursortasten bewegen.

```
260 CLS
270 PRINT STRING$(&50,"_")
280 PRINT"                    T   E   X   T       e   r   s   t   e   l   l
  e  n"
290 PRINT STRING$(&50,"_")
300 LOCATE 10,10:INPUT"Wie soll der Text heissen ";textname$
310 CLS
320 zeile=1: spalte=1
330 PLOT spalte*8+8,400-zeile*16+8,1
340 a$=INKEY$:IF a$="" GOTO 330
350 PLOT spalte*8+8,400-zeile*16+8,0
360 IF a$=CHR$(13) GOTO 470
370 IF a$=CHR$(93) GOTO 630
380 IF a$=CHR$(240) GOTO 510
390 IF a$=CHR$(241) GOTO 540
400 IF a$=CHR$(242) GOTO 570
410 IF a$=CHR$(243) GOTO 600
420 spalte=spalte+1
430 IF spalte=80 THEN GOSUB 650
440 inhalt$(spalte,zeile)=a$
450 LOCATE spalte,zeile:PRINT a$
460 GOTO 330
470 spalte=1
480 zeile=zeile+1
490 IF zeile>24 THEN zeile=24
500 GOTO 330
510 zeile=zeile-1
520 IF zeile<1 THEN zeile=1
530 GOTO 330
540 zeile=zeile+1
550 IF zeile>24 THEN zeile=24
560 GOTO 330
```

```
570 IF spalte=1 THEN zeile=zeile-1:spalte=80
580 spalte=spalte-1
590 GOTO 330
600 IF spalte=80 THEN zeile=zeile+1:spalte=1
610 spalte=spalte+1
620 GOTO 330
630 inhalt$(spalte,zeile)=CHR$(93)
640 GOTO 100
650 zeile=zeile+1:spalte=1
660 IF zeile>24 THEN zeile=24
670 GOTO 330
680 CLS
```

Erläuterung zum Programmteil:

260	Bildschirm löschen.
270-290	Ausgabe des Programmteilkopfes.
300	Eingabe des Textnamens.
310	Löschen des Bildschirmes.
320	Die Positionsvariablen "zeile" und "spalte" werden auf eins gesetzt, somit befindet sich der Cursor in der oberen linken Ecke des Bildschirms. Diese Variablen geben von nun an immer die aktuelle Cursorposition wieder.
330	Mit diesem Befehl wird der Cursor (ein kleiner Graphikpunkt) auf die Position der nächsten Bildschirmstelle gebracht.
340	Ein Zeichen wird von der Tastatur eingelesen.
350	Hier wird der alte Cursor gelöscht,da er nicht mehr aktuell ist, dies geschieht nur mittels des letzten Arguments, der "0".
360-410	Hier werden der Reihe nach folgende Tasten abgefragt: RETURN-Taste, ECKIGE KLAMMER ZU, Cursortasten. Wurde eine dieser 6 Tasten gedrückt, verzweigt das Programm zu einer entsprechenden Routine.
420	Hier wird der Wert für die Variable "spalte" um eins erhöht, um deren Wert auf die aktuelle Position zu bringen.

430	Ist die letzte Position in dieser Zeile erreicht (spalte=80), wird zu einer Routine verzweigt.
440	Das Zeichen, das in a$ eingelesen wurde, wird in die Variable inhalt$ überführt, die mit der Position des Zeichens (spalte,zeile) ausgestattet ist.
450	Hier wird das Zeichen selbst an der richtigen Stelle auf dem Bildschirm ausgegeben.
460	Rücksprung, um das nächste Zeichen einzulesen
470-500	Routine zur Behandlung der RETURN-Taste. Wurde diese gedrückt, springt der Cursor an den Anfang der nächsten Zeile, indem der Wert für "zeile" um eins erhöht wird, und "spalte" auf eins gesetzt wird. Zeile 490 sorgt dafür, daß nicht aus dem Bildschirm gesprungen wird.
510-530	Routine zur Behandlung der Cursor-Up-Taste. Wurde diese gedrückt, wird der Wert der Variablen "zeile" einfach um eins heruntergesetzt. Zeile 520 sorgt wieder dafür, daß der Cursor nicht aus dem Bild herausgeht.
540-560	Routine zur Behandlung der Cursor-Down-Taste. Wurde diese Taste gedrückt, wird der Wert von "zeile" um eins erhöht. Zeile 550 ist wieder die entsprechende Abfangroutine.
570-590	Routine zur Behandlung der Cursor-Left-Taste. Wurde diese Taste gedrückt, wird zuerst abgefragt, ob der Cursor am Anfang der Zeile war. Wenn ja, wird der Wert für "zeile" um eins herabgesetzt und "spalte" auf 80 gesetzt. Ist dieses nicht der Fall, wird "spalte" einfach um eins herabgesetzt.
600-620	Routine zur Behandlung der Cursor-Right-Taste. Wurde diese Taste gedrückt, wird zuerst abgefragt, ob der Cursor am Ende der Zeile war. Wenn ja, wird der Wert für "zeile" um eins erhöht und "spalte" auf eins gesetzt. Ist dieses nicht der Fall, wird "spalte" einfach um eins erhöht.

630-640 Routine zur Behandlung der Taste "ECKIGE KLAMMER ZU". Zuerst wird an der letzten Cursorposition ein Zeichen eingefügt, um später das Textende lokalisieren zu können. Dann wird zum Menü gesprungen.

650-670 Routine für den Fall, daß beim Schreiben das Zeilenende erreicht wurde. "zeile" wird hier um eins erhöht und "spalte" auf eins gesetzt. Zeile 660 ist wieder eine Abfangroutine.

Verwendete Variablen:
----

textname$	Enthält den Namen des Textes.
zeile	Enthält die aktuelle Zeilennummer des Cursors.
spalte	Enthält die aktuelle Spalte des Cursors.
a$	Enthält das zuletzt eingegebene Zeichen.
inhalt$ (spalte, zeile)	Enthält den ganzen Text, jeder Buchstabe ist definiert durch seine Spalten- und Zeilennummer.

Verbesserungsmöglichkeiten:

Die Eingaberoutine selbst braucht eigentlich nicht verbessert zu werden, da Sie wirklich optimal ist. Ein möglicher Verbesserungspunkt wäre, mehr Tasten abzufragen. Also zum Beispiel noch die DEL und die CLR Taste oder eine Taste, die den ganzen Text und Bildschirm löscht.

Ein Punkt, der zu verbessern wäre, ist die Länge, die der Text haben kann. Zur Zeit ist diese ja auf 24 Zeilen

beschränkt. In der Eingaberoutine ist dieses relativ einfach zu erreichen, indem die Variable inhalt$ größer dimensioniert wird und alle Parameter, die jetzt auf 24 zeigen, auf einen entsprechend höheren Wert heraufgesetzt werden (beachten Sie aber hierbei bitte, daß Sie den Speicher nicht überlasten). Schwerer wird es erst bei der Änderungsroutine sowie dem Programmpunkt "TEXT ansehen", dieses kann allerdings von einem etwas geübteren Programmierer bewerkstelligt werden.

Bedienung des Programmteiles:

Nachdem Sie diesen Teil angewählt haben, müssen Sie zuerst den Namen des Textes eingeben. Das kann zum Beispiel sein : Geschäftsbrief vom 03.08.1984, Brief an Monika und Frank oder ähnliches. Haben Sie den Namen eingegeben und danach die ENTER-Taste betätigt, befinden Sie sich im "Eingabeblatt". Hier können Sie nun Ihren Text eingeben. Dieses geschieht einfach durch die entsprechenden Tasten. Drücken Sie die ENTER-Taste, so springt der Cursor zum Anfang der nächsten Zeile. Mit Hilfe der sogenannten CURSOR-TASTEN können Sie den Cursor frei über den Bildschirm bewegen und dort dann weiter schreiben. Sie brauchen also nicht nur mit der ENTER- und SPACEtaste zu arbeiten.

Haben Sie Ihren Text eingegeben, drücken Sie die Taste ECKIGE-KLAMMER-ZU. Das Programm springt dann zum Menü zurück.

Verwenden Sie bitte nie die Taste ECKIGE-KLAMMER-AUF. Diese haben wir als EOF (=end of file, deutsch: Ende des Textes) -Zeichen genommen. Erkennt eine Routine des CPC-Textes dieses Zeichen, bedeutet das: Hier ist der Text zu Ende. Das Symbol wird automatisch vom Programm an den Text angehängt, wenn Sie den Programmteil Text erstellen verlassen.

Einige andere Tasten, wie die DEL- oder CLR- Taste, haben zur Zeit keine Funktion, können aber, wenn sie im Zusammenspiel mit CAPS LOCK gedrückt werden, undefinierte Zeichen ergeben. Dieses ist aber nicht weiter tragisch, da Sie dann mit den CURSOR-Tasten an die entsprechende Stelle "fahren" können, um dann das Zeichen entweder mittels der SPACE-Taste zu löschen, oder Sie überschreiben es mit einem anderen Zeichen.

T E I L 3 - T E X T A N S E H E N -

Der folgende Teil ist eine einfache, aber unbedingt nötige Routine.

Sie dient dazu, den im Speicher befindlichen Text auf dem Bildschirm auszugeben. Dieses ist zum Beispiel nötig. wenn Sie gerade einen Text eingeladen haben, und Sie wissen möchten, welcher es ist oder ob Änderungen nötig sind.

Natürlich läßt sich diese Routine für vieles andere verwenden. Wie wäre es, wenn Sie den Monitor in das Schaufenster Ihres Geschäftes stellen. Vorher haben Sie einen entsprechenden Werbetext erstellt, den Sie dann mittels nachfolgendem Programmteil ständig auf dem Monitor stehen lassen können.

```
690 PRINT STRING$(&50,"_")
700 PRINT"                    T  E  X  T        a  n  s  e  h
  e  n"
710 PRINT STRING$(&50,"_")
720 LOCATE 10,10:PRINT"Zum Text betrachten bitte eine Taste
druecken"
730 CALL &BB18
740 CLS
750 FOR zeile=1 TO 24
760 FOR spalte=1 TO 80
770 a$=inhalt$(spalte,zeile)
780 IF a$="" THEN a$=CHR$(32)
790 IF a$=CHR$(93) THEN GOTO 830
795 inhalt$(spalte,zeile)=a$
800 LOCATE spalte,zeile:PRINT a$
810 NEXT spalte
820 NEXT zeile
830 CALL &BB18
840 GOTO 100
```

Erläuterung zum Programm:

680	Löschen des Bildschirms.
690-710	Ausgabe des Programmteilkopfes.
720	Anweisung an den Benutzer.
730	Routine, die das Programm so lange anhält, bis eine Taste gedrückt wird. Wir haben diese Routine schon in einer der vorigen Kapitel aufgeführt. Sie sollten sie ruhig öfter benutzen, da sie wirklich sehr praktisch ist. Sie sollten Sie weniger verwenden, wenn Sie Ihre auf dem CPC erstellten Programme auch auf anderen Computern lauffähig machen möchten, da diese die Routine nicht, oder unter einer anderen Adresse haben.
740	Löschen des Bildschirmes.
750-760	Eröffnung der Schleife, um jede Position des Bildschirmes anzusprechen.
770	Der Variablen a$ wird nacheinander der ganze Text zugewiesen.
780	Abfrage, ob die Bildschirmstelle leer ist (unter "leer" wird hier verstanden, daß es auch kein SPACE ist). Wenn ja, wird ein Leerzeichen ausgegeben.
790	Abfrage, ob das Zeichen = EOF (Ende des Textes) ist. Wenn ja, wird die weitere Ausgabe übersprungen.
795	Das Leerzeichen wird in den Text eingebunden
800	Ausgabe des aktuellen Zeichens an der richtigen Position.
810-820	Schließen der Schleife.
830	Durch diesen Befehl bleibt der Bildschirm so lange erhalten, bis Sie keine Taste drücken. Haben Sie eine Taste gedrückt, springt das Programm zum Menü zurück.
840	Rücksprung zum Menü.

Verwendete Variablen:
----

zeile	Bekannt
spalte	Bekannt
a$	Bekannt
inhalt$	Bekannt

Verbesserungsmöglichkeiten:

Grundlegend sind keine Verbesserungen nötig. Vielleicht gestalten Sie das Programm noch etwas umfangreicher, indem Sie die Möglichkeit einbauen, sich nur einen bestimmten Teil des Bildschirmes ausgeben zu lassen. Dieses ist relativ einfach, da Sie die Werte der Variablen "zeile" und "spalte" auf den entsprechenden Wert setzen können.

Bedienung des Programmteiles:

Nachdem Sie den Teil im Menü angewählt haben, brauchen Sie bei der entsprechenden Aufforderung durch das Programm nur eine Taste drücken. Danch wird dann Ihr Text auf dem Bildschirm ausgegeben. Dieses dauert leider etwas länger, da jedes Zeichen einzeln ausgegeben werden muß. Ist der Text dann fertig ausgegeben, bleibt er auf dem Bildschirm. Möchten Sie ins Menü zurück, brauchen Sie einfach nur irgendeine Taste zu drücken.

Denken Sie bitte daran, daß Sie diese Routine immer anspringen, nachdem Sie einen Text erstellt haben und bevor Sie irgendeine andere Funktion auslösen.

TEIL 4 -TEXT ÄNDERN-

Der nachfolgende Programmteil ist gedacht, um Fehler in Ihrem Text zu beseitigen.
Wir haben für Sie hier auch ein kleines Lehrstück im Fach Programmieren eingebaut, auf das wir im Teil Erläuterungen ausführlich hinweisen. Achten Sie hierauf bitte besonders gut.

```
850 CLS
860 PRINT STRING$(&50,"_")
870 PRINT"                    T  E  X  T          a  e  n  d
e  r  n"
880 PRINT STRING$(&50,"_")
890 LOCATE 10,10:PRINT"Zum Text aendern bitte eine Taste dru
ecken"
900 CALL &BB18
910 CLS
920 FOR zeile=1 TO 24
930 FOR spalte=1 TO 80
940 a$=inhalt$(spalte,zeile)
950 IF a$="" THEN a$=CHR$(32)
960 IF a$=CHR$(91) THEN GOTO 1000
970 LOCATE spalte,zeile:PRINT a$
980 NEXT spalte
990 NEXT zeile
1000 GOTO 320
```

Erläuterung zum Programm:

850	Löschen des Bildschirmes.
860-880	Ausgabe des Programmteilkopfes.
890	Anweisung an den Benutzer.
900	Programm wartet, bis Taste gedrückt wird
910	Löschen des Bildschirmes.
920-990	Routine zur Ausgabe des Textes auf dem Bildschirm. Diese Routine wurde im vorigen Teil ausführlich erläutert. Sehen Sie hier bitte noch einmal nach, wenn Sie nicht mehr ganz orientiert sein sollten.
1000	Sprung zur Eingaberoutine des Programmteiles TEXT ERSTELLEN. Dieses ist ein raffinierter Trick, um Speicherplatz zu sparen. Da Sie mit der Routine ja schreiben können, und man ändern auch als Überschreiben verstehen kann, besteht kein Grund, diesen programmtechnischen Trick nicht anzuwenden. Auf solche Möglichkeiten sollten Sie immer achten, b.z.w. Ihr Programm so konzipieren, daß Sie Routinen erstellen, die Sie von verschiedenen Teilen des Programmes benutzen können. Es spart zum einen wichtigen Speicherplatz, den Sie dann für andere Daten nutzen können, zum anderen verkürzt es die Programmentwicklungszeit, da Sie weniger an Zeilen zu schreiben haben und somit auch weniger Fehler beseitigen müssen (dieses ist der zeitintensivste Teil bei der Erstellung eines Programms). Diese Technik wird vor allem von Profis benutzt, da für diese bekanntlich Zeit gleich Geld ist.

Verwendete Variablen:
----

zeile	Bekannt
spalte	Bekannt
a$	Bekannt
inhalt$	Bekannt

Variablen der Routine aus dem Programmteil TEXT ERSTELLEN

Verbesserungsmöglichkeiten:

Verbesserungsmöglichkeiten sind hier auch wieder nur im Bereich des erhöhten Komforts nötig. So wäre es sehr nützlich, wenn man Leerstellen einfügen könnte, um diese später mit Zeichen zu füllen. Oder wie wäre es, wenn Sie ganze Wörter oder Zeilen löschen, b.z.w. einfügen könnten. Dieses ist durch die relativ einfache Ablage aller Zeichen in einer einzigen Variablen, die nur durch 2 Werte indiziert ist, nicht allzu schwer. Allerdings sollten Sie sich nicht als ganz blutiger Anfänger an diese Aufgabe setzen, da Sie sonst am Ende vielleicht zu enttäuscht wären, wenn Sie es nicht schaffen. Aber auch für den geübten Programmierer gilt: Nehmen Sie Änderungen nur vor, wenn Sie den Aufbau des Programms voll verstanden haben, was Ihnen allerdings nach entsprechendem Studium aller Ausführung ein Leichtes sein sollte.

Bedienung des Programmteiles:

Wenn Sie den Programmteil angewählt haben, drücken Sie bitte zuerst eine Taste. Danach gibt Ihnen dann das Programm den im Speicher befindlichen Text auf dem Bildschirm aus. Sie

können genauso arbeiten, wie im Teil "TEXT ERSTELLEN". Haben Sie eine Stelle gefunden, wo Sie einen Tippfehler gemacht haben, "fahren" Sie mit Hilfe der Cursortasten dort hin und überschreiben diese Stelle einfach. Verlassen können Sie den Programmteil mit der ECKIGEN-KLAMMER-ZU. Es gilt auch alles im Teil "TEXT EINGEBEN" gesagte, für diesen Teil. Bitte beachten Sie insbesondere die Funktion der Taste ECKIGE-KLAMMER-AUF.

Der Umgang mit dieser Routine bedarf einiger Übung. Erstellen Sie also ruhig einmal einen Text mit extra vielen Fehlern und verbessern Sie diese dann. Das ist eine der besten Methoden, um den Umgang mit dieser Programmroutine zu erlernen.

T E I L 5   - T E X T L Ö S C H E N -

Der nachfolgende Teil dient dazu, Text zu löschen. Dieses ist zum Beispiel nötig, wenn Sie einen Teil geschrieben haben, Sie beim Lesen aber feststellen, daß man den Sachverhalt beim besten Willen so nicht formulieren kann. Dann ist es nötig, den im Speicher befindlichen Text zu löschen.

```
1010 CLS
1020 PRINT STRING$(&50,"_")
1030 PRINT"                    T  E  X  T       l  o  e  s  c
  h  e  n"
1040 PRINT STRING$(&50,"_")
1050 LOCATE 10,10:INPUT"Moechten Sie den Text wirklich loesc
hen  (j/n)";wahl$
1060 IF wahl$="j" THEN CLEAR:GOTO 1070
1065 GOTO 100
1070 RUN
```

Erläuterung zum Programm:

1010	Löschen des Bildschirmes.
1020-1040	Ausgabe des Programmteilkopfes.
1050	Hier müssen Sie Ihre Entscheidung noch einmal bestätigen, da Sie den Programmteil aus Versehen angewählt haben könnten.
1060	Wenn die Variable wahl$ den Wert "j" enthält, wird der Text mittels des CPC-Befehls CLEAR gelöscht. Das Programm springt nach 1070.
1065	Hier arbeitet das Programm weiter, wenn Sie Ihren Text nicht löschen möchten. Das Programm springt zum Menü zurück.
1070	Hierhin springt das Programm, wenn der Text gelöscht wurde. Das Programm wird mittels RUN gestartet, damit die Dimensionierung der Variablen inhalt$ wiederhergestellt wird, da diese beim CLEAR verlorengeht.

Verwendete Variablen:

---

wahl$ Bekannt

Verbesserungsmöglichkeiten:

Hier sind Verbesserungen eigentlich nur optischer Natur, da dieser Programmteil, wie einige andere auch, programm- technisch keine Anforderung stellt.

Die einzige brauchbare Verbesserungsmöglichkeit wäre es, einzelne Blöcke zu löschen. Diese Erweiterung sollte aber eigentlich im Programmteil ÄNDERN eingebaut werden.

Bedienung des Programmteiles:

Die Bedienung dieses Teiles ist wieder denkbar einfach. Sie müssen sich nur entscheiden, ob Sie den Text wirklich löschen möchten oder nicht. Für den Fall, daß der Text gelöscht werden soll, geben Sie ein "j" ein, sonst ein nein. Den Rest erledigt das Programm für Sie. Sie befinden sich auf jeden Fall hinterher wieder im Menü.

T E I L 6   - T E X T   A B S P E I C H E R N -

Dieser Teil dient dazu, Ihre Texte auf Band zu bringen. Dieses ist ein ganz wichtiger Baustein einer Textverarbeitung, da er den Hauptvorzugsgrund einer solchen, gegenüber der Schreibmaschine hat. Warum?
Sie können einmal erstellte Texte immer wieder laden, ändern, ausdrucken. Dieses ist bei einer Schreibmaschine (nomalerweise) nicht möglich, es sei denn, Sie hätten eine sogenannte Speicherschreibmaschine. Allerdings werden Sie sich wohl bei einem Preivergleich eher für den CPC 464 + CPC-TEXT aus diesem Buch entscheiden.

```
1080 CLS
1090 PRINT STRING$(&50,"_")
1100 PRINT"          T  E  X  T    a  b  s  p  e  i  c  h
 e  r  n"
1110 PRINT STRING$(&50,"_")
1120 LOCATE 10,10:PRINT"Zum abspeichern bitte eine Taste dru
ecken"
1130 CALL &BB18
1150 OPENOUT "
1160 FOR zeilen=1 TO 24
1170 FOR spalten=1 TO 80
1180 a$=inhalt$(spalten,zeilen)
1190 PRINT#9,a$
1200 NEXT spalten
1210 NEXT zeilen
1220 CLOSEOUT
1230 GOTO 100
```

Erläuterung zum Programm:

1080	Löschen des Bildschirmes.
1090-1100	Ausgabe des Programmteilkopfes.
1120	Anweisung an den Benutzer.
1130	Programm wartet, bis eine Taste gedrückt wurde.
1140	Schreibgeschwindigkeit wird heraufgesetzt, damit das Abspeichern nicht zu viel Zeit erfordert. Verwenden Sie bitte nur gute

Kassetten, da es sonst durch die hohe Aufzeichnungsdichte später zu Lesefehlern kommen könnte, was heißt: Ihr Text ist verloren! Möchten Sie nicht so teure Kassetten verwenden, lassen Sie diesen Befehl beim abtippen einfach heraus. Trotzdem sollten Sie nie zu schlechte Kassetten verwenden, allerdings auch keine superteuren Chromdioxid-Kassetten, da wir die Erfahrung gemacht haben, daß diese nicht das optimale Bandmaterial sind. Wählen Sie am besten eine 60er Kassette für 3-5 DM. Ideal sind natürlich spezielle Computerkassetten, die auch kürzere Bänder haben (C-10 oder C-15). So müssen Sie nicht so viel spulen. Eins sollten Sie aber immer beachten: Erstellen Sie in nicht zu großen Abständen Sicherheitskopien von wichtigen Texten, da auch das beste Band irgendeinmal versagt.

1150	Eröffnen einer Aufzeichnungsdatei unter dem Namen des Textes
1160-1170	Eröffnen der Schleife, um jede Position des Bildschirmes anzuprechen.
1180	Der Variablen a$ wird der Reihe nach der Inhalt jeder Bildschirmposition zugewiesen.
1190	Der Inhalt der Variablen wird auf Band geschrieben.
1200-1210	Schließen der Schleife.
1220	Schließen der Ausgabedatei

Verwendete Variablen:
----

textname$	Bekannt
zeilen	Bekannt
spalten	Bekannt
a$	Bekannt
inhalt$	Bekannt

Verbesserungsmöglichkeiten:

Es sind in diesem Programmteil aus programmtechnischer Sicht wieder nicht nötig, da sich an der Speicherform nichts ändern läßt. Allerdings wäre es möglich, die Übertragungsrate durch eine Eingabe im Programm einstellen zu lassen.Außerdem wäre es möglich, nach dem Speichervorgang beim Benutzer nachzufragen, ob der Text noch einmal abgespeichert werden soll.

Bedienung des Programmteiles:

Nachdem Sie diesen Teil angewählt haben, müssen Sie zuerst eine Taste drücken.Danach eröffnet das Programm eine Datei zum Schreiben auf Band, dieses dauert einige Zeit. Danach werden Sie aufgefordert, die PLAY sowie die RECORD-Taste auf dem Bandgerät zu drücken. Haben Sie dieses ausgeführt, betätigen Sie eine Taste und Ihr Text wird abgespeichert. Danach kehrt das Programm zum Menü zurück.

TEIL 7 -TEXT LADEN-

Dieser Teil ist das Äquivalent zum vorhergehenden Programmteil.
Hiermit können Sie die Texte, die Sie mit dem CPC-Text geschrieben und abgespeichert haben, wieder einladen.

```
1240 CLS
1250 PRINT STRING$(&50,"_")
1260 PRINT"                              T  E  X  T     l  a  d
 e  n"
1270 PRINT STRING$(&50,"_")
1280 LOCATE 10,10:PRINT"Zum laden bitte eine taste druecken"
1290 CALL &BB18
1300 OPENIN "
1310 FOR zeilen=1 TO 24
1320 FOR spalten=1 TO 80
1330 INPUT #9,a$
1340 inhalt$(spalten,zeilen)=a$
1350 NEXT spalten
1360 NEXT zeilen
1370 CLOSEIN
1380 GOTO 100
```

Erläuterung zum Programm:

1240	Löschen des Bildschirmes.
1250-1270	Ausgabe des Programmteilkopfes.
1280	Anweisung an den Benutzer.
1290	Programm wartet, bis eine Taste gedrückt wurde.
1300	Eröffnen einer Datei zum Lesen von Band.
1310-1320	Eröffnung der Schleifen zum Einlesen der Daten.
1330	Einlesen der Daten in die Variable a$.
1340	Der Inhalt wird der Variablen inhalt$. zugewiesen, mit der entsprechenden Position auf dem Bildschirm zugewiesen.
1350-1360	Ende der Schleifen.
1370	Schließen der Einlesedatei.
1380	Rücksprung zum Menü.

Verwendete Variablen:

zeilen	Bekannt
spalten	Bekannt
inhalt$	Bekannt
a$	Bekannt

Verbesserungsmöglichkeiten:

Auch hier sind Verbesserungen nicht nötig. Die beiden Verbesserungen, die wir beim letzten Teil ansprachen, treffen ja auf den Teil LADEN nicht zu, da Ihr CPC selbstständig die Übertragungsrate erkennt und Sie diese nicht einzustellen brauchen. Ein zweites mal denselben Text zu laden, würde ja auch wenig Sinn machen. Dieses könnte höchstens bei Ladefehlern nützlich sein. Allerdings sollten diese so selten auftreten, daß eine extra Routine die dieses Berücksichtigt, nicht nötig sein sollte.

Bedienung des Programmteiles:

Die Bedienung ist eigentlich analog zu der im Teil TEXT LADEN, nur daß Sie hier nicht zusätzlich RECORD drücken müssen, sondern nur die PLAY-Taste.

T E I L 8 - T E X T A U S D R U C K E N -

Nachfolgender Teil hat die Aufgabe, Ihren Text zu Papier zu bringen. Dieses ist nötig, da es meistens nicht möglich ist, "elektronische" Post zu verschicken. Deshalb müssen Sie Ihre Texte ausdrucken können, damit Sie dann diese verschicken können.

```
1390 CLS
1400 PRINT STRING$(&50,"_")
1410 PRINT"                         T E X T     d r u c k
 e n"
1420 PRINT STRING$(&50,"_")
1430 LOCATE 10,10:PRINT"Zum ausdrucken bitte eine taste drue
cken"
1440 CALL &BB18
1450 FOR zeilen=1 TO 24
1460 FOR spalten=1 TO 80
1470 a$=inhalt$(spalten,zeilen)
1480 PRINT #8,a$;
1490 NEXT spalten
1495 PRINT #8,CHR$(13);CHR$(10);
1500 NEXT zeilen
1530 GOTO 100
```

Erläuterung zum Programm:

1390	Löschen des Bildschirmes.
1400-1420	Ausgabe des Programmteilkopfes.
1430	Anweisung an den Benutzer.
1440	Programm wartet, bis Taste gedrückt wird.
1450-1460	Eröffnen der Schleifen zur Ausgabe auf dem Drucker.
1470	Der Variablen a$ wird nacheinander der Inhalt des Bildschirmes zugewiesen
1480	Ausgabe des Zeichens auf dem Drucker
1490-1500	Schließen der Schleifen.
1495	Nach jeder Zeile wird der Druckkopf auf den Anfang gestellt.
1530	Rücksprung zum Menü.

Verwendete Variablen:
----

zeilen	Bekannt
spalten	Bekannt
inhalt$	Bekannt
a$	Bekannt
wahl$	Bekannt

Verbesserungsmöglichkeiten:

Wir haben auch hier, wie beim CPC-DATA, nur eine sehr einfache Ausgaberoutine verwandt. Bei unserer Version wird der Text so ausgegeben, wie Sie ihn auf dem Bildschirm erstellt haben.

Eine gute Druckroutine sollte dem Benutzer aber die Möglichkeit geben, eine formatierte Ausgabe vorzunehmen.

Wie sieht eine formatierte Ausgabe aus? Denkbar wäre zum Beispiel, daß Sie nach Wahl Ihren Text rechts- und/oder linksbündig ausgeben können. Dieses bedeutet, daß die Zwischenräume in den Zeilen soweit vergrößert werden, bis die Wörter mit dem Zeilenanfang beginnen und mit diesem auch genau enden (bündig abschließen). Der Text, den Sie gerade lesen wurde so erstellt.

Das ist allerings nicht ganz einfach zu programmieren und vor allem in Basic relativ langsam. Diese Erweiterung sollte also wirklich nur vom geübten Programmierer vorgenommen werden, am besten in Maschinensprache.

Bedienung des Programmteiles:

Nachdem Sie sich im Teil TEXT DRUCKEN befinden, müssen Sie den Drucker anschließen und dann eine Taste drücken. Den Rest erledigt wieder das Programmm für Sie.
Danach werden Sie aufgefordert, einzugeben, ob Sie noch einen Ausdruck angefertigt haben möchten. Wenn Sie noch einen möchten, geben Sie ein "j" ein, sonst ein "n".
Die Druckanpassung wird in Zeile 1495 vorgenommen.

TEIL 9 -PROGRAMM BEENDEN-

Der nachfolgende Programmteil hat die Aufgabe, das Programm zu beenden.

Natürlich können Sie dieses auch mit Hilfe der BREAK-Taste machen, aber so ist es eleganter.

```
1540 CLS
1550 PRINT STRING$(&50,"_")
1560 PRINT"              P  R  O  G  R  A  M  M    b  e  e  n
d  e  n"
1570 PRINT STRING$(&50,"_")
1580 LOCATE 5,10:INPUT"Moechten Sie das Programm wirklich be
enden (j/n)";wahl$
1590 IF wahl$="j" THEN END ELSE GOTO 100
```

Erläuterung zum Programm:

1540	Löschen des Bildschirmes.
1550-1570	Ausgabe des Programmteilkopfes.
1580	Rückfrage, ob Sie das Programm wirklich beenden möchten für den Fall, daß Sie diesen Programmteil aus Versehen angewählt haben.
1590	Wenn wahl$ gleich "j" ist, wird das Programm beendet, andernfalls wird zum Menü zurückgesprungen.

Verwendete Variablen:
----

wahl$	Bekannt

Verbesserungsmöglichkeiten:

Wie in vielen Programmteilen, so sind auch in diesem Verbesserungen nur im optischen Bereich sinnvoll.

Bedienung des Programmteiles:

Die Bedienung ist hier wieder denkbar einfach, somit natürlich optimal für den Anfänger. Sie müssen nur ein "j" eingeben, wenn Sie das Programm beenden möchten. Ansonsten geben Sie ein "n" ein und das Programm springt zum Menü zurück.

Bedenken Sie bitte, daß, wenn Sie das Programm beendet haben, alle Daten, die Sie nicht zuvor Kasette gesichert haben, unwiederbringlich verloren sind.

7.4 FANG DIE BOMBE

Da Sie sicher nicht nur an Ihrem CPC hart arbeiten wollen, haben wir uns gedacht, ein oder zwei kleine Spiele mit in dieses Buch zu bringen. Unser erstes Spiel heißt "Fang die Bombe" und wird Ihnen auf den ersten Blick durch seine Kürze recht unproblematisch vorkommen, jedoch ist es nicht ganz so einfach zu spielen, wie es zuerst den Anschein hat.

Es geht darum, mit einem "Wagen" am unteren Bildschirmrand herabfallende Bomben aufzufangen. Fangen Sie die Bombe, so erhalten Sie einen Punkt, fällt sie auf den Boden, bekommen Sie einen Punkt abgezogen. Die Fallgeschwindigkeit und die Anzahl der Bomben können Sie zu Anfang des Spieles bestimmen.
Bewegt wird der Wagen mit den beiden Cursortasten für links und rechts.

Wichtig beim Abtippen des Programms ist es, daß in Zeile 180 zwei Leerstellen nach dem PRINT Befehl eingegeben werden.

Die beste Anfangsgeschwindigkeit ist 0.5. Mit dieser Geschwindigkeit dürfte es Ihnen keine Schwierigkeiten bereiten, die Bomben zu fangen. Sollten Sie jedoch die Gefahr lieben, so können Sie ja eine Geschwindigkeit größer 2 wählen.

Vermeiden Sie es besser das Programm mittels "ESC" zu verlassen, da wir die Wiederholfunktion der Tasten herunter gesetzt haben, und es Ihnen danach Schwierigkeiten bereiten wird, einen Befehl einzugeben. Bei einem kurzen Tastendruck wird das Zeichen mehrere Male auf dem Bildschirm ausgegeben wird.

```
10 REM FANG DIE BOMBE
20 MODE 1
30 SYMBOL AFTER 57
40 SYMBOL 58,128,192,192,255,255,201,28,8
50 SYMBOL 59,1,3,3,255,255,147,58,16
60 SYMBOL 60,4,8,28,62,127,127,62,28
70 SYMBOL 61,255,227,255,255,255,255,255,255
80 INPUT "fallgeschwindigkeit";fa
90 INPUT"Anzahl der Bomben";bo
100 SPEED KEY 1,1
110 CLS
120 ko=10
130 FOR bomben=1 TO bo
140 CLS
150 LOCATE 1,22:PRINT STRING$(&28,"=")
160 d=1:anf=INT(RND(1)*20)+1
170 LOCATE anf,d:PRINT" "
180 LOCATE ko,21:PRINT":;"
190 LOCATE anf,d+fa:PRINT"<"
200 LOCATE ko,21:PRINT"  "
210 d=d+fa
220 IF INT (d)>=20 THEN GOTO 280
230 in$=INKEY$
240 ko=ko+(1 AND IN$=CHR$(243))-(1 AND IN$=CHR$(242))
250 IF ko<1 THEN ko=1
260 IF ko>25 THEN ko=25
270 GOTO 170
280 IF ko=anf OR anf=ko+1 THEN tr=tr+1
290 IF ko<>anf AND ko<>anf-1  THEN tr=tr-1
300 NEXT
310 CLS
320 SPEED KEY 20,3
330 LOCATE 15,10:PRINT"PUNKTE:",tr
340 LOCATE 10,20:PRINT"noch einmal?"
350 INPUT a$
360 IF a$="j" THEN RUN
370 SPEED KEY 20,3
```

Noch ein guter Tip für ganz mutige Leser: Ändern Sie die Zeile 140 in:

```
140 D=1:ANF=INT(RND(1)*30)
```

Ändern Sie danach noch Zeile 240 in:

```
240 IF KO: 30 THEN KO=30
```

Nach diesen Änderungen kommen die Bomben auf noch weiterer horizontaler Breite geflogen, und es ist noch schwieriger, sie zu fangen.

Sie können das Programm natürlich auch noch an weiteren Stellen ausbauen. So könnten Sie die Graphik des Hintergrundes verbessern, oder Ton einbauen, oder wie wäre es, wenn Sie ein paar selbstdefinierte Zeichen für die aufschlagende Bombe entwickeln, und diese Zeichen mittels des LOCATE- Befehls über den Bildschirm bewegen, wenn die Bombe nicht gefangen wurde?

7.5. SCHIFFE VERSENKEN

Nachfolgend finden Sie das zweite von uns für Sie kreierte Spiel.
Lassen Sie sich nicht von dem Namen abschrecken, es ist nicht die 126. Version des bekannten Spieles.

Bei unserem SCHIFFE VERSENKEN ist der Spielsinn ein anderer.

Im ersten Teil des Programms haben Sie einige Parameter einzugeben.

Zuerst die Schwierigkeitsstufe. Diese liegt im Bereich von 0-2. Bei den Schwierigkeitsstufen haben wir eine Eigenschaft des CPC ausgenutzt, die verschiedenen Zeichenmodi (20-40-80). In der Stufe 0 haben wir mit dem 20-Zeichenschirm gearbeitet. Hier sieht die Graphik am besten aus, da die x-Ausdehnung eines Zeichens hier relativ groß ist. Natürlich ist der Schwierigkeitsgrad hier am geringsten. Die anderen Schwierigkeitsstufen sind ebenfalls analog zu den Zeichenmodi.

Der zweite einzugebende Parameter ist die Zeiteinheit. Je kleiner diese gewählt wurde, desto weniger Zeit steht Ihnen für Ihre Aufgabe zur Verfügung. Wir gehen von einem Richtwert, von einer Einheit aus, natürlich ist auch weniger möglich.

Der dritte Parameter ist die Bombengeschwindigkeit. Mit Hilfe dieses Wertes können Sie die Geschwindigkeit einstellen, mit der sich die Bombe über den Bildschirm bewegt.

Die letzte Eingabe ist die Anzahl der Bomben, mit denen Sie spielen möchten.

Nur haben wir Sie so lange auf die Folter gespannt, daß wir Ihnen langsam das Spiel selbst erklären sollten.

Haben Sie die oben beschriebenen Werte eingegeben, erscheint das eigentliche Spielbild. Dort sehen Sie ein großes Schiff, mit einer Brücke und einer Kanone am Bug, außerdem ein U-Boot, sowie eine Bombe neben diesem. Ihre Aufgabe ist es nun, mit Hilfe der Cursortasten die Bombe auf den Bug des Schiffes zu lenken und dieses zu zerstören. Hierfür bekommen Sie Punkte, sollten Sie das Schiff nicht in der von Ihnen angegeben Zeit treffen, zeigt das Programm das nächste Spielbild.
Am Ende des Spiels erhalten Sie eine Wertung Ihrer Leistung. In diese fließt die Anzahl der Treffer, die Zeit die, Ihnen zur Verfügung stand, sowie der Modus und die Bombengeschwindigkeit ein.

Natürlich können Sie dieses Spiel nach Ihren Wünschen mit Farbe und Ton austatten. Auch das sollte Ihnen nach Lektüre der Kapitel 2 und 3 dieses Buches keine besonderen Schwierigkeiten mehr bereiten.

```
10 MODE 2
15 treffer=0
20 SYMBOL AFTER 32
30 LOCATE 20,10:PRINT"S C H I F F E   V E R S E N K E N"
40 GOSUB 350
50 LOCATE 1,15:INPUT"Welche Schwierigkeitsstufe (0-2)";a
60 INPUT"Wieviele Zeiteinheiten zur Bombensteuerung";b
70 INPUT"Geschwindigkeit der Bombe";gesch
80 INPUT"Wieviele Bomben";bomben
85 DIM bo1(bomben)
90 MODE a
95 FOR bb=1 TO bomben
96 CLS
100 IF a=0 THEN f=18
110 IF a=1 THEN f=38
120 IF a=2 THEN f=78
130 yp=INT(RND(1)*24)+2
140 xp=INT(RND(1)*f)+1
150 yp2=INT(RND(1)*25)+1
160 xp2=INT(RND(1)*f)+1
170 IF xp=xp2 OR yp=yp2 THEN GOTO 130
171 xp3=xp2+2:yp3=yp2
172 b1=b*100
175 SPEED KEY 1,gesch
176 FOR mnn=1 TO b1
180 LOCATE xp+2,yp-1:PRINT"!"
190 LOCATE xp,yp:PRINT"&%#"
200 LOCATE xp2,yp2:PRINT"][" 
220 LOCATE xp3,yp3:PRINT" "
230 in$=INKEY$
240 xp3=xp3+(1 AND in$=CHR$(243))-(1 AND in$=CHR$(242))
250 yp3=yp3+(1 AND in$=CHR$(241))-(1 AND in$=CHR$(240))
260 IF xp3<1 THEN xp3=1
270 IF xp3>f+2 THEN xp3=f+2
280 IF yp3<1 THEN yp3=1
290 IF yp3>25 THEN yp3=25
300 IF yp3=25 AND xp3=f+2 THEN xp3=f+1
310 LOCATE xp3,yp3:PRINT"^"
320 IF xp3=xp AND yp3=yp THEN GOTO 440
330 NEXT mnn
335 NEXT bb
```

```
340 GOTO 3000
350 REM zeichen definieren
360 SYMBOL 33,62,8,28,28,124,124,124,252
370 SYMBOL 35,252,252,252,255,254,252,248,240
380 SYMBOL 37,1,129,129,255,255,255,255,255
390 SYMBOL 38,126,15,15,127,63,15,3,1
400 SYMBOL 91,128,192,252,254,255,255,254,0
410 SYMBOL 93,0,1,31,63,255,255,63,0
420 SYMBOL 94,0,0,156,190,127,190,156,0
430 RETURN
440 MODE 0
445 treffer=treffer+1
446 LOCATE 3,12
447 bo1(bb)=((b1-mnn)/b1)*10
448 FOR zei=1 TO 15
449 PRINT CHR$(224);
450 NEXT
451 LOCATE 3,13:PRINT CHR$(224);"T R E F F E R";CHR$(224)
452 LOCATE 3,14
453 FOR zei=1 TO 15
454 PRINT CHR$(224);
455 NEXT
460 SOUND 129,450,150,15,,,15
465 FOR xx=1 TO 3000
466 NEXT xx
467 MODE a
470 GOTO 335
3000 MODE 2
3005 FOR II=1 TO BOMBEN
3006 PUNKTE=PUNKTE+BO1(II)
3007 NEXT
3008 PUNKTE=PUNKTE*TREFFER*(A+1)
3010 LOCATE 15,9:PRINT"S P I E L    Z U    E N D E"
3020 LOCATE 2,11:PRINT"Sie haben bei";bomben;"Bomben";treffe
r;"Treffer gehabt."
3021 LOCATE 2,13:PRINT "DAS IST EINE PUNKTZAHL VON:";PUNKTE
3025 SPEED KEY 20,3
3030 INPUT"Moechten Sie noch einmal spielen";wahl$
3040 IF wahl$="j" THEN RUN
3050 END
```

DAS STEHT DRIN:

CPC464 Tips & Tricks bietet eine hochinteressante Sammlung von Anregungen, Ideen und fertigen Lösungen zur Programmierung und Anwendung Ihres CPC464.

Aus dem Inhalt:

- Hardwareaufbau
- Betriebssystem, Basic-Tokens
- Bildschirmaufbau
- Anwendungen der Window-Möglichkeiten
- Eine komplette Dateiverwaltung
- Ein gut dokumentiertes Textverarbeitungsprogramm
- Soundeditor
- Komfortabler Zeichengenerator
- Spannende Spiele
- Maschinensprache-Monitor

. . . und vieles mehr

UND GESCHRIEBEN HABEN DIESES BUCH:

Lothar Englisch ist bekannter Autor verschiedener DATA BECKER-Bücher (Das Maschinensprachebuch zum C-64). Jörg Germer ist Schüler und sein Hobby ist natürlich der Computer. Thomas Scheuse ist ebenfalls Gymnasiast mit vier Jahren Erfahrung am Homecomputer. Frank Thrun ist Programmierer in der DATA BECKER-Softwareabteilung.

ISBN 3-89011-039-8